SCORE: Similarity-Aware Contextual Overlap-Redundancy Eviction for Efficient KV Cache Compression in LLMs *

Gilha Lee*[1], Seungil Lee*[2], Hyun Kim†[1],

[1] Department of Electrical and Information Engineering, Research Center for EIT
Seoul National University of Science and Technology, Seoul, Korea
[2] Korea Electronics Technology Institute, Seongnam, Gyeonggi-do, Korea
gilha1@seoultech.ac.kr, seungil66@keti.re.kr, hyunkim@seoultech.ac.kr

Abstract

Recent advances in large language models (LLMs) have unlocked remarkable long-context capabilities, enabling breakthroughs across diverse NLP tasks. However, despite architectural progress, the key-value (KV) cache remains a critical memory bottleneck during inference. Prior work has explored cache optimization via eviction strategies, yet most rely on heuristic or single-axis importance metrics, neglecting the nuanced and dynamic interplay between layers and attention heads. In this paper, we propose *SCORE* (Similarity-aware Contextual Overlap-Redundancy Eviction), a novel framework that introduces a distance-based multi-level similarity metric to quantify and eliminate structural redundancy within the KV cache. By dynamically reallocating cache budgets across layers and heads and employing a redundancy-aware greedy token selection mechanism, *SCORE* preserves semantic diversity while minimizing memory overhead. Extensive experiments on long-context benchmarks such as LongBench and NeedleBench show that *SCORE* retains 96.2% of full KV cache performance on Llama-3-8B averaged over LongBench using only 1.5% of the full cache, consistently outperforming state-of-the-art baselines under strict memory constraints. These results underscore the value of fine-grained, context-aware cache management for scalable and efficient long-context inference in LLMs.

1 Introduction

Large language models (LLMs) have shown exceptional long-context understanding, achieving state-of-the-art performance across a wide range of natural language processing (NLP) tasks, including multi-turn dialogue, document summarization, and information retrieval (Zhao et al., 2023). Recent models, such as GPT-4 (Achiam et al., 2023), Claude 3.5 (Anthropic, 2024), LLaMA 3.1 (Grattafiori et al., 2024), and Mistral (Jiang et al., 2023), demonstrate significantly improved long-context capabilities, with some supporting up to one million tokens (Anthropic, 2024), enabling the stable processing of substantially extended input sequences. Moreover, recent advances have further accelerated efforts to deploy LLMs in resource-constrained environments, including on-device settings (Kwon et al., 2023; Xu et al., 2024; Li et al., 2024b). However, the key-value (KV) cache remains a major memory bottleneck for long-context processing, significantly limiting efficiency.

Accordingly, recent research has focused on optimizing the KV cache without altering model architecture (Dao, 2023; Acharya et al., 2024). A key direction is the eviction of low-importance KV pairs to reduce memory usage (Ge et al., 2023; Zhang et al., 2023; Xiao et al., 2023), typically guided by attention scores or heuristic rules. However, as shown in (Dong et al., 2021; Zhang et al., 2022), the contribution of layers and attention heads to text generation is highly uneven and context-dependent. Efficient cache utilization thus requires fine-grained, dynamic strategies that account for this variability. To address such non-uniformity, prior work has explored both layer-level and head-level budgeting. For example, PyramidKV (Cai et al., 2024) allocates cache budgets based on information density across layers, whereas Cake (Qin et al., 2025) employs a cascading mechanism to adjust budgets during prefill, as illustrated in Figure 1(a). More fine-grained methods like SnapKV (Li et al., 2024c) and HeadKV (Fu et al., 2024) estimate head-level importance and reallocate budgets accordingly, as shown in Figure 1(b). However, these methods rely primarily on simple statistics—such as entropy or variance—without leveraging richer signals (***L1***). Moreover, they fail to jointly capture redundancy across layer–head interactions, which further limits

*Equal Contribution. †Corresponding Author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 49020–49038
October 24-29, 2026 ©2026 Association for Computational Linguistics

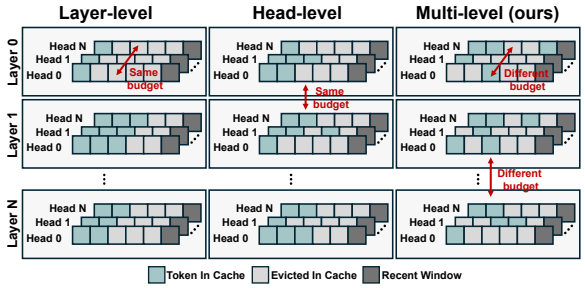


Figure 1: KV cache eviction strategies: (a) Layer-wise allocation assigns distinct budgets across layers; (b) Head-wise allocation enables finer-grained control; (c) SCORE combines both to reduce redundancy and optimize cache usage.

their effectiveness (***L2***). In addition, their simplistic budget allocation often leads to limited token diversity, resulting in poor coverage and making it difficult to capture the overall context (***L3***).

To mitigate these limitations, we introduce **S**imilarity-aware **C**ontextual **O**verlap-**R**edundancy **E**viction (*SCORE*), a cache management framework that quantifies representational redundancy across layers and heads using a distance-based multi-level similarity metric, as illustrated in Figure 1(c). *SCORE* reduces functionally redundant cache entries while preserving contextual diversity through dynamic budget reallocation and selective cache retention. This design captures representational divergence and contextual progression more effectively than prior heuristics-based methods, enhancing redundancy-aware information preservation. Experiments show that *SCORE* maintains 96.2% of full-cache accuracy on Llama-3-8B averaged over LongBench at a cache budget of 128 (1.5% of the full cache), consistently outperforming prior eviction-based methods on long-context benchmarks.

The main contributions of *SCORE* are as follows: (i) *SCORE* introduces distance-based metrics to explicitly model relational redundancy across layers and heads in the KV cache (for ***L1***). (ii) **Redundancy-aware multi-level metric.** To capture hierarchical information flow and representational diversity, we introduce multi-level similarity metrics that quantify redundancy across layers and heads in the KV cache (for ***L2***). (iii) **Hierarchical budget allocation.** The proposed *SCORE* framework dynamically reallocates cache budgets across layers and heads based on redundancy scores, prioritizing informative and non-redundant tokens under constrained memory. (iv) **Greedy token selection.** Our redundancy-aware, greedy token selection algorithm maximizes information diversity in the cache by accounting for similarity (for ***L3***).

2 Backgrounds

2.1 Inference Operations with KV Cache

Inference in transformer-based LLMs typically consists of two stages: a prefill stage that encodes the input sequence, and a decode stage that sequentially generates output tokens.

Prefill stage: Given an input prompt tensor $X \in \mathbb{R}^{S\times D}$, where S and D denote the sequence length and the hidden dimension, respectively, the key and value representations are computed as follows:

$$Q = XW_Q,\ K = XW_K,\ V = XW_V, \quad (1)$$

where $W_Q, W_K, W_V \in \mathbb{R}^{D\times D}$ are the learnable projection matrices that map the input sequence to query, key, and value representations, respectively. The resulting KV tensors are then stored in the KV cache, where they are reused during the subsequent decode stage to avoid redundant computation.

Decode stage: During decoding, for each newly generated token $x_i \in \mathbb{R}^{1\times D}$ at time step i, the corresponding key and value are computed, while previous information is retrieved from the KV cache. The new key and value are then appended to the existing cache along the sequence dimension via concatenation, as follows:

$$K \leftarrow K \cup x_i W_K,\ \ V \leftarrow V \cup x_i W_V. \quad (2)$$

Then, the current query $q_i = x_i W_Q$ computes attention weights via a scaled dot-product with the full K, and aggregates V accordingly to produce the output $x_{i,\text{out}}$ as follows:

$$x_{i,\text{out}} = \text{Softmax}\left(\frac{q_i K^\top}{\sqrt{D}}\right) V. \quad (3)$$

As decoding progresses, the size of the KV cache increases linearly (*e.g.*, 62.5GB in Llama3-8B-Instruct (Shi et al., 2024)), which becomes a major bottleneck in terms of memory usage and latency. Efficient cache compression and management are therefore essential for long-context processing.

2.2 KV Cache Eviction

H2O (Zhang et al., 2023) improves efficiency by focusing on important tokens (heavy hitters), while ROCO (Ren and Zhu, 2024) and Scissorhands (Liu et al., 2023) similarly retain key tokens based on attention scores. StreamingLLM (Xiao et al., 2023)

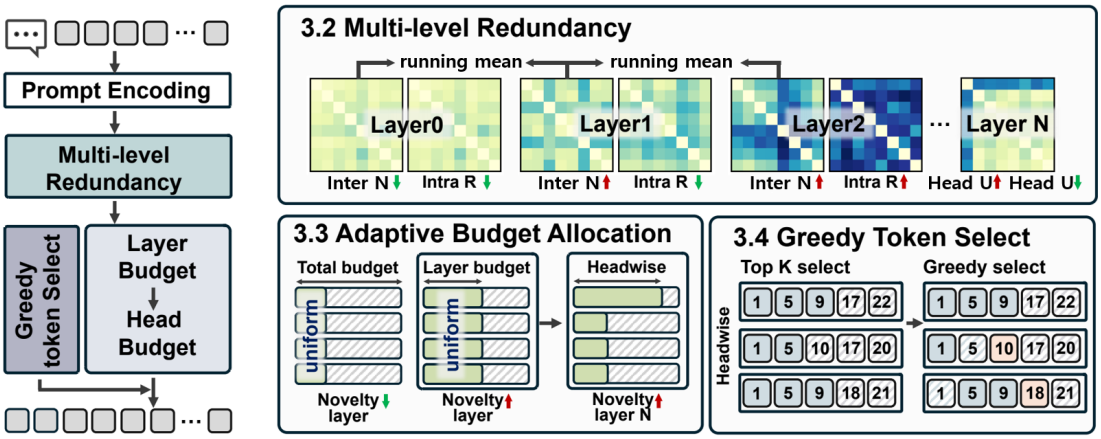


Figure 2: An overview of SCORE. The model computes redundancy scores across layers and heads, which guide a dynamic budget allocation strategy and inform redundancy-aware token selection to achieve diverse representation. For each layer: left shows temporal redundancy with the previous layer; right shows intra-layer head similarity.

and LM-Infinite (Han et al., 2023) prioritize nearby tokens relevant to generation, but uniform eviction often leads to information loss. To address this, dynamic budget allocation methods have emerged. PyramidKV (Cai et al., 2024) employs a pyramidal attention pattern, allocating more cache to lower layers while summarizing higher-layer information. CAKE (Qin et al., 2025) analyzes attention dispersion and temporal shifts to reassign cache budgets in a cascading manner. However, both rely solely on layer-level budgeting. To enable finer granularity, SnapKV (Li et al., 2024c) clusters attention distributions at the head level, and HeadKV (Fu et al., 2024) introduces an allocation scheme based on retrieval and reasoning contributions to minimize post-eviction degradation. While these methods achieve budgeting at either the layer or head level (***L2***), they fail to jointly consider both dimensions and rely heavily on attention scores (***L1***), which limits their ability to capture cross-head redundancy. Moreover, even with well-determined budgets, token-level diversity is overlooked (***L3***), leading to overlapping selections and limited coverage. To overcome these limitations, we propose *SCORE*, a unified framework that models relational redundancy via distance-based similarity and integrates it into both budget allocation and token eviction.

3 Proposed Method

3.1 Overall Architecture of SCORE

As illustrated in Figure 2, *SCORE* is a redundancy-aware KV cache management framework. It consists of three core components: a multi-level redundancy scoring module (Section 3.2) that quantifies representational redundancy across layers and heads; a dynamic budget allocation strategy (Section 3.3) that selectively retains KV entries based on their relative importance; and a selection mechanism (Section 3.4) that prioritizes diverse and informative tokens for cache retention. Algorithm 1 in Appendix F summarizes the end-to-end pipeline of *SCORE*.

3.2 Multi-Level Redundancy Scoring

According to prior studies (Michel et al., 2019; Dalvi et al., 2020), many layers and attention heads in transformer models often learn functionally similar representations, leading to structural redundancy. To quantitatively assess such redundancy and diversity, we propose multi-level evaluation metrics based on inter-head similarity and mitigate ***L1***, ***L2***. Specifically, we sample k token positions from each head and compute the average pairwise distance of their attention scores. Given a cosine distance function $\delta(\cdot,\cdot)$ that measures the dissimilarity between two attention score vectors, the pairwise distance between head i and j is defined as:

$$D_{(i,j)} = \frac{1}{k^2}\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}\delta(\mathbf{a}_i^{(s)},\ \mathbf{a}_j^{(t)}), \qquad (4)$$

where $\mathbf{a}_i^{(s)} \in \mathbb{R}^n$ denotes the attention score vector of head i at the s-th sampled token for indices $i, j \in \{0, 1, \ldots, (L \cdot H - 1)\}$. Figure 3(a) visualizes the distance matrix D across all attention heads and layers. The matrix shows that redundancy is not uniformly distributed but is often concentrated in specific layers or heads, motivating the multi-level analysis metrics introduced below. All metrics derived from D are directionally consistent:

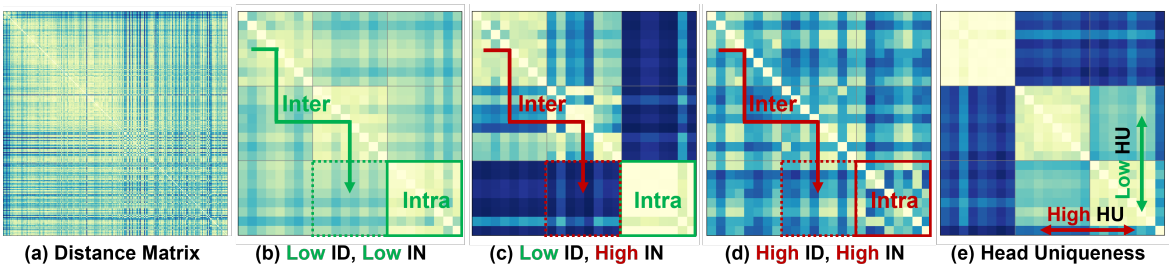


Figure 3: Head-wise cosine distance matrices. From left to right: (a) the full distance matrix across all heads and layers; (b)–(d) three examples of 3×3 layer-wise distance matrices, each showing different levels of intra-layer redundancy and inter-layer diversity; (e) an illustrative example highlighting head-level redundancy. Brighter colors indicate higher similarity.

lower values denote higher redundancy, whereas higher values reflect greater diversity, novelty, or uniqueness across heads.

Intra-layer Diversity. To analyze the extent of diverse representations within each layer, we define the intra-layer diversity ($\mathcal{ID}$) of layer l as the average distance across its attention heads, computed by evaluating a distance measure D over all pairs of heads (h, h') within the same layer:

$$\mathcal{ID}_l = \frac{1}{H^2} \sum_{h,h'=0}^{H-1} D_{lH+h,\, lH+h'}. \tag{5}$$

Here, lower $\mathcal{ID}_l$ indicates higher redundancy due to increased similarity across heads, whereas higher $\mathcal{ID}_l$ suggests greater functional diversity.

Inter-layer Novelty. To quantify the degree of novel information introduced relative to the preceding layer, we introduce the inter-layer novelty ($\mathcal{IN}$) metric. It captures shifts across depth by comparing the distance of adjacent layers against an accumulated baseline. For layer l, $\mathcal{IN}_l$ is defined as:

$$\mathcal{IN}_l = \frac{|\mu_l - E_{l-1}|}{\mu_l + \epsilon}, \tag{6}$$

where μ_l denotes the average distance between layer l and its previous layer, computed using the inter-layer distance matrix D. For the first layer, we set $\mathcal{IN}_0 = \mathcal{ID}_0$. The accumulated baseline E_{l-1} tracks the distance trend across depth and is updated using a running mean:

$$E_l = \frac{l}{l+1} E_{l-1} + \frac{1}{l+1} \mu_l. \tag{7}$$

Higher $\mathcal{IN}_l$ indicates that layer l produces more novel and distinctive outputs compared to the accumulated patterns of preceding layers. Figure 2 (specifically, inside the box corresponding 3.2) illustrates this by comparing the left temporal matrices across layers, where notable increases in deviation reveal points of significant representational change. Figures 3(b)-(d) present varied examples with different levels of redundancy and temporal variation, clarifying the hierarchical structure and functional differentiation.

Head Uniqueness. Beyond analyzing diversity at the layer level, we introduce the head uniqueness ($\mathcal{HU}$) metric to quantitatively assess the uniqueness of individual attention heads. This metric quantifies how distinct a head's attention pattern is compared to others in the same and preceding layers. An illustrative example is shown in Figure 3(e). For head h at layer l, $\mathcal{HU}_{l,h}$ is defined as follows:

$$\mathcal{HU}_{l,h} = \frac{1}{2H} \sum_{i=0}^{1} \sum_{h'=0}^{H-1} D_{lH+h,\, (l-i)H+h'}. \tag{8}$$

Here, lower $\mathcal{HU}_{l,h}$ indicates that the head is more redundant with its neighboring heads, whereas higher $\mathcal{HU}_{l,h}$ indicates that the head is more distinguishable and likely to fulfill a unique functional role. The metric complements layer-level averages by enabling fine-grained evaluation of diversity and redundancy at the head level.

3.3 Hierarchical Budget Allocation

Layer budget allocation. We build on the cascading strategy from (Qin et al., 2025) and introduce adaptive budget reallocation driven by the diversity and novelty of hierarchical representations. At each stage, only interactions between active layers and their immediate predecessors are considered. Accordingly, the pairwise distance matrix D is computed only between adjacent layers. Cache budgets are dynamically assigned based on a layer priority score $\mathcal{N}_l$, which integrates intra-layer diversity and inter-layer novelty as follows:

$$\mathcal{N}_l = \mathcal{ID}_l^{\lambda_1} \cdot \mathcal{IN}_l^{\lambda_2}. \tag{9}$$

where λ_1 and λ_2 are weighting factors that control the relative importance of intra-layer diversity and inter-layer novelty. The total cache budget B_{total} is then allocated proportionally to the layer priority scores:

$$B_l^{(m)} = \frac{\mathcal{N}_l}{\sum_{k=0}^{L-1} \mathcal{N}_k} \cdot B_{\text{total}}, B_l^{(m)} < B_l^{(m-1)}, \tag{10}$$

At cascading stage m, the cache budget for layer l is assigned in proportion to its priority score, enabling adaptive allocation based on importance.

Head budget allocation. As shown in Figure 2, cache budgets are first allocated at the layer level and then further distributed across heads. To account for variations in importance and uniqueness across heads, we propose a fine-grained head-wise allocation strategy. We adopt a lightweight importance estimation based on the accumulated attention score matrix $A_l \in \mathbb{R}^{H \times S}$. Specifically, for each head h, we quantify its contribution using the attention allocation magnitude:

$$M_{l,h} = \sum_{s=1}^{S} A_l[h, s]. \tag{11}$$

This quantity reflects how strongly each head participates in the attention distribution at layer l. To incorporate both the attention-based importance and distance-based uniqueness of each head, we combine $M_{l,h}$ with the head uniqueness score $\mathcal{HU}_{l,h}$ when deriving the head-wise budget:

$$B_{l,h} = B_l \cdot \frac{HU_{l,h} \cdot M_{l,h}}{\sum_{h'=0}^{H-1} (HU_{l,h'} \cdot M_{l,h'})}. \tag{12}$$

This strategy refines layer-level cache allocation by jointly considering head-wise importance and redundancy, effectively addressing ***L2*** and preserving distinctive information. It serves as a key mechanism for improving eviction performance in LLMs.

3.4 Greedy Token Selection

The multi-level budget allocation strategy improves cache efficiency by assessing redundancy and contribution at both the layer and head levels. However, even with fine-grained allocation, redundant heads often select overlapping tokens. As illustrated by the gray tokens attended by multiple heads in the 'Top-K select' example of Figure 2, different heads frequently focus on the same token positions, resulting in redundant cache usage.

To mitigate head-level concentration, we introduce a redundancy-aware soft selection mechanism based on per-(head, token) dependency rather than aggregate attention. We define the dependency of head h on token s as the share of its attention relative to all heads in the layer:

$$U_{l,h,s} = \frac{A_l[h, s]}{\sum_{h'=0}^{H-1} A_l[h', s] + \epsilon}, \tag{13}$$

where $U_{l,h,s} \to 1$ indicates that the token is essential to head h, while $U_{l,h,s} \to 1/H$ indicates uniform attention across heads. We then apply a single multiplicative penalty that jointly considers two redundancy signals:

$$\widetilde{A}_l[h, s] = A_l[h, s] \cdot \exp^{-(1-U_{l,h,s})(1-\mathcal{HU}_{l,h})}. \tag{14}$$

The product $(1 - U_{l,h,s})(1 - \mathcal{HU}_{l,h})$ implements a redundancy-aware multiplicative penalty, whose strength increases when the head depends less on the token and is more redundant within the layer. The penalty is attenuated when either condition is weak, thereby protecting tokens strongly preferred by individual heads. This generalizes the single-head argmax exemption used in prior heuristics to every head that strongly depends on a token, while preserving layer-level head differentiation through $\mathcal{HU}_{l,h}$. It promotes complementary evidence across heads, effectively addressing the ***L3*** issue with minimal modification to existing token selection pipelines.

4 Experiments and Analysis

4.1 Experimental Setup

Models & Tasks. We evaluate representative open-source LLMs with context lengths from 4K to 128K tokens, including multi-head attention models (Llama2-7B & 13B (Touvron et al., 2023)) and grouped-query attention models (Llama3-8B-Instruct & Llama3-70B (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Qwen2.5-7B-Instruct (Yang et al., 2024)). Performance under compressed KV cache is measured on LongBench (Bai et al., 2023) & LongBench V2 (Bai et al., 2024) and InfiniteBench (Zhang et al., 2024), with NeedleBench (Li et al., 2024a) and Reasoning-in-a-Haystack (Kuratov et al., 2024) for fine-grained retrieval and long-range reasoning. We further evaluate Qwen2.5-7B-Instruct-1M to validate *SCORE* in longer-context settings.

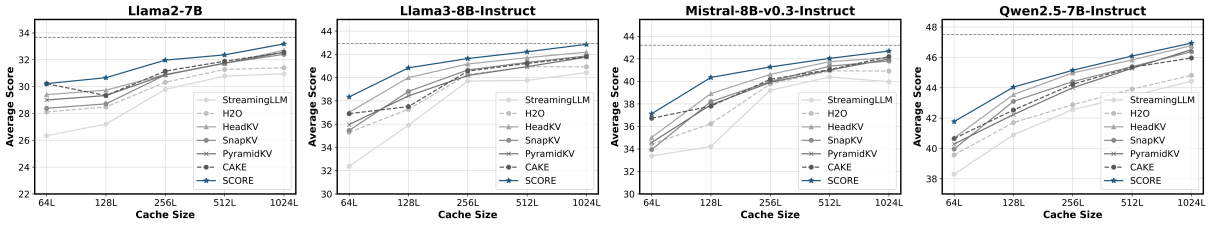


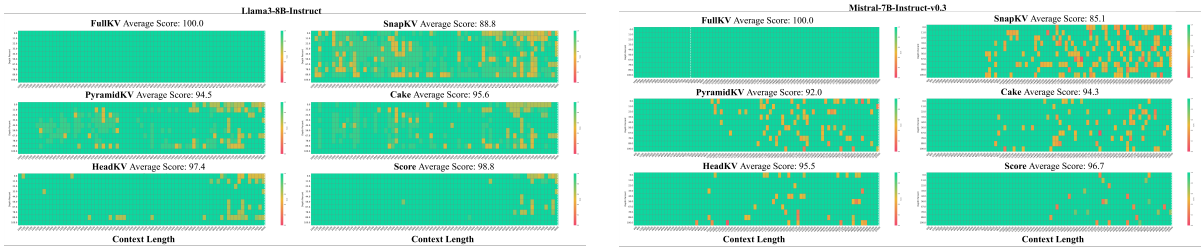
Figure 4: Mean performance across 16 LongBench datasets for varying KV cache sizes. The dashed line indicates the full cache performance.

	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	
Llama2-7B	Full KV	18.39	21.13	35.54	31.35	25.61	10.64	25.57	22.18	26.26	65.50	89.41	41.03	6.00	8.50	58.67	53.00	33.67
	StreamingLLM	11.27	15.93	28.45	24.77	20.92	6.46	15.42	16.80	18.97	60.55	79.42	35.32	3.00	2.00	50.82	45.22	27.21
	H2O	11.82	17.75	29.70	24.98	21.95	8.33	16.98	18.49	20.49	62.50	80.79	36.78	3.50	3.00	51.95	46.68	28.48
	HeadKV	14.86	19.25	28.40	27.39	24.30	8.38	18.29	20.01	21.10	63.50	81.99	37.92	4.00	4.50	53.54	48.34	29.74
	SnapKV	14.12	18.50	27.25	28.45	21.83	7.94	17.23	20.04	20.30	58.00	81.85	37.43	4.00	4.00	52.05	46.41	28.71
	PyramidKV	15.20	18.70	29.02	29.72	23.57	7.95	17.87	20.01	20.32	60.50	82.59	36.71	4.00	4.00	52.58	46.68	29.34
	CAKE	15.62	19.01	32.10	29.46	24.78	9.48	18.75	20.01	**22.52**	63.50	82.95	38.75	4.50	6.00	54.59	48.66	30.67
	SCORE(ours)	**16.43**	**20.06**	**32.52**	**30.61**	**25.19**	**10.54**	**19.31**	**20.62**	22.41	**64.00**	**83.00**	**38.92**	**5.50**	**7.00**	**54.25**	**49.23**	**31.22**
Llama3-8B -Instruct	Full KV	25.56	39.43	45.23	45.37	35.65	21.63	28.63	23.35	26.81	74.00	90.48	42.52	4.80	69.25	56.97	52.42	42.63
	StreamingLLM	20.58	28.66	25.17	37.45	18.72	18.87	19.01	20.17	18.81	62.00	88.57	38.96	3.51	68.00	55.42	50.86	35.91
	H2O	21.90	30.11	26.55	38.95	20.01	20.29	20.37	21.58	20.18	63.50	89.96	**40.39**	4.40	69.25	56.83	52.30	37.29
	HeadKV	22.47	30.13	40.38	44.90	31.06	21.10	20.70	22.31	21.91	71.00	**90.82**	39.62	4.35	**69.50**	57.65	52.65	40.03
	SnapKV	22.17	28.96	36.29	42.10	29.25	19.78	20.11	22.56	21.46	66.00	89.72	38.89	4.50	69.00	57.24	**52.97**	38.81
	PyramidKV	22.10	26.94	36.86	40.38	29.42	16.34	20.34	**22.70**	21.99	67.00	89.35	39.77	4.50	69.00	56.55	51.67	38.43
	CAKE	22.10	32.19	34.52	39.06	30.45	20.72	20.40	21.85	20.98	46.00	89.64	39.74	4.50	**69.50**	56.46	52.14	37.52
	SCORE(ours)	**24.64**	**33.33**	**40.82**	**45.50**	**34.36**	**21.55**	**21.53**	22.66	**22.95**	**72.50**	90.76	40.29	**5.00**	**69.50**	**57.70**	52.86	**41.00**
Mistral-7B -Instruction-v0.3	Full KV	29.53	41.58	53.13	49.22	39.51	28.58	34.68	26.42	27.82	80.50	92.14	47.44	5.50	98.00	58.45	59.54	48.25
	StreamingLLM	22.83	25.80	41.54	39.11	23.73	15.44	18.04	16.34	15.86	45.55	83.82	37.55	3.00	75.00	43.54	40.46	34.23
	H2O	23.29	27.23	42.08	40.68	24.86	16.48	18.75	17.65	17.07	56.50	85.03	38.88	4.50	80.00	44.53	42.44	36.25
	HeadKV	26.04	30.25	47.93	43.17	**32.13**	22.63	20.73	18.24	18.79	65.50	85.29	40.69	5.00	76.50	44.93	44.95	38.92
	SnapKV	25.72	28.56	46.34	43.52	29.10	20.86	19.51	18.09	18.49	64.50	84.81	40.20	**5.50**	78.50	43.99	43.68	38.21
	PyramidKV	25.50	27.47	46.23	44.02	30.05	21.10	20.08	18.15	18.10	63.50	85.02	40.37	4.50	78.50	43.52	40.73	37.93
	CAKE	26.17	27.26	43.73	41.30	25.89	17.30	20.22	18.01	17.65	57.50	85.53	39.11	5.00	82.00	45.89	43.39	37.25
	SCORE(ours)	**26.85**	**32.69**	**50.53**	**44.52**	32.01	**23.91**	**21.53**	**19.91**	**19.99**	**69.50**	**86.41**	**41.31**	4.50	**83.00**	**46.62**	**46.63**	**40.65**
Qwen2.5-7B -Instruct	Full KV	28.96	43.63	50.84	54.50	47.06	30.71	32.23	21.42	22.25	72.00	89.31	45.84	4.03	98.42	60.54	66.68	48.03
	StreamingLLM	20.98	32.71	42.34	50.21	39.17	23.86	27.99	17.33	15.97	68.50	79.07	41.26	1.46	88.67	52.55	52.11	40.89
	H2O	21.91	33.95	43.03	51.26	40.01	24.44	29.73	17.74	16.39	69.50	79.79	41.72	1.87	90.22	53.08	52.61	41.70
	HeadKV	22.38	34.28	44.45	51.89	40.95	26.60	29.63	18.79	17.97	70.00	81.20	42.41	3.61	97.48	53.21	54.79	43.10
	SnapKV	22.87	34.75	44.55	52.04	40.43	25.70	29.93	19.69	17.98	**71.50**	80.35	43.65	3.19	80.25	54.75	54.07	42.23
	PyramidKV	22.58	34.94	43.04	51.98	40.96	23.13	29.97	19.56	17.52	**71.50**	80.49	43.08	3.61	88.13	55.49	54.44	42.53
	CAKE	23.64	35.11	44.83	52.33	**41.87**	27.89	29.15	19.39	18.47	71.00	82.06	43.93	3.54	95.75	54.73	53.16	43.55
	SCORE(ours)	**24.41**	**35.71**	**44.84**	**52.76**	41.69	**28.08**	**30.23**	**19.79**	**18.98**	71.00	**83.16**	**44.44**	**3.95**	**98.12**	**55.73**	**55.13**	**44.25**

Table 1: Performance comparison over 16 datasets from LongBench. Results are measured with a cache size of 128. The best score is marked in **bold**, and the second best is marked with underline.

4.2 Main Results

Evaluation on LongBench. We evaluate *SCORE* on LongBench, which includes 16 long-context tasks spanning diverse capabilities. As shown in Figure 4, *SCORE* consistently outperforms prior eviction methods across KV cache budgets, with the advantage becoming more pronounced under tighter budgets ($B_{\text{total}} \leq 128\text{L}$). This trend indicates that explicitly exploiting redundancy across layers and heads becomes increasingly beneficial as the available cache decreases. At a cache budget of 128, Table 1 further shows that *SCORE* achieves the best performance on most tasks across Llama2-7B, Llama3-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen2.5-7B-Instruct. On LLaMA3-8B-Instruct, *SCORE* averages 41.00, outperforming HeadKV by $+0.97$ while retaining 96.2% of full-cache performance with only 1.5% of the cache. Similar gains on Mistral-7B-Instruct-v0.3 and Qwen2.5-7B-Instruct demonstrate that the improvement is consistent across different model families. Notably, the advantage persists across both MHA- and GQA-based architectures, indicating that the proposed allocation strategy is not tied to a particular attention configuration. Consistent gains across cache budgets further show that *SCORE* remains effective across compression levels rather than at a single operating point. In particular, *SCORE* shows strong gains on multi-document QA and synthetic tasks, where information must be preserved across distant context positions. Together, the results in Figure 4 and Table 1 show that *SCORE* provides a favorable accuracy–memory trade-off, particularly in the low-cache regime targeted by KV cache compression. Full results across cache sizes are provided in Appendix H.



(a) Llama-3-8B-Instruct

(b) Mistral-7B-Instruct-v0.3

Figure 5: NeedleBench test results with 128L. SCORE significantly outperforms all strong baselines on both models.

	Method	Single-Doc.	Multi-Doc.	Summ.	Few-shot	Synthetic	Code	Avg.
Llama2-13B	Full KV	27.43	28.27	24.09	62.76	8.50	52.33	33.90
	HeadKV	17.17	26.57	21.07	64.52	8.91	41.21	29.91
	SnapKV	16.10	27.16	20.44	63.38	9.88	39.54	29.42
	PyramidKV	17.06	27.05	20.53	63.43	9.88	40.39	29.72
	CAKE	16.97	26.91	21.00	64.95	9.63	39.39	29.81
	SCORE(ours)	**18.86**	**27.98**	**22.24**	**66.48**	**10.05**	**44.89**	**31.75**
Llama3-70B	Full KV	41.33	47.94	24.68	72.42	37.52	74.40	49.72
	HeadKV	37.88	46.75	18.49	70.26	37.18	68.46	46.50
	SnapKV	36.15	46.68	18.45	69.60	36.96	68.23	46.01
	PyramidKV	36.90	46.89	19.28	69.87	36.36	68.66	46.33
	CAKE	38.05	46.79	18.94	70.73	37.14	**68.78**	46.74
	SCORE(ours)	**39.55**	**47.01**	**19.93**	**71.63**	**37.25**	68.62	**47.33**

Table 2: Scalability analysis on LongBench (cache size 128). The best score is marked in **bold**, and the second best is marked with underline.

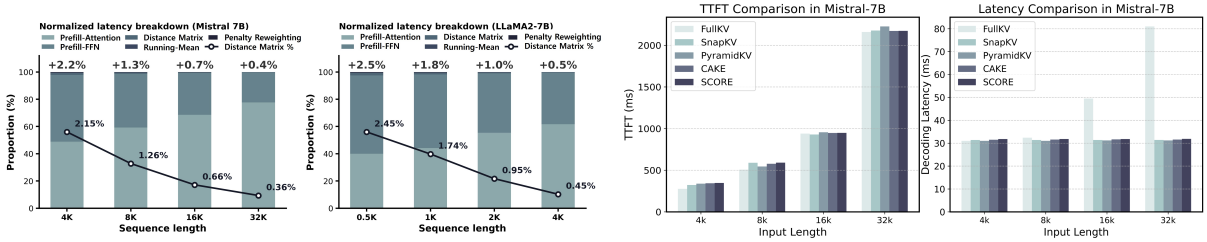
Method	Easy	Hard	Short	Medium	Long	Avg.
Full KV	31.25	25.08	34.44	24.19	22.22	27.44
Llama3-8B-Instruct, Cache size=128						
PyramidKV	29.81	22.45	31.97	22.48	19.45	25.23
CAKE	29.27	21.13	**32.98**	21.22	20.18	24.96
HeadKV	29.53	22.74	32.25	22.97	20.10	25.52
SCORE	**29.99**	**23.22**	32.87	**23.75**	**20.35**	**26.04**
Llama3-8B-Instruct, Cache size=1024						
PyramidKV	30.21	23.13	32.75	23.45	21.22	26.15
CAKE	29.50	21.31	33.25	22.61	21.06	25.55
HeadKV	30.22	23.97	33.24	23.23	20.14	26.16
SCORE	**30.30**	**24.15**	**33.91**	**24.20**	**21.45**	**26.80**

Table 3: Performance comparison on Llama3-8B-Instruct over LongBench V2. Results are measured with 128 and 1024 cache sizes

NeedleBench Results. To assess the retrieval capability of *SCORE*, we evaluate on NeedleBench using LLaMA-3-8B-Instruct and Mistral-7B-Instruct-v0.3, with maximum context lengths of 8K and 32K tokens, respectively (Figure 5). At a cache size of 128, *SCORE* demonstrates strong retrieval fidelity with minimal degradation as context length increases. Notably, on Mistral-7B-Instruct-v0.3, *SCORE* achieves an accuracy of 96.7, outperforming HeadKV (95.5) by $+1.2$ points, while consistently matching or exceeding existing methods on Llama3-8B-Instruct. *SCORE* maintains robust retrieval under severe cache compression with only negligible degradation compared to FullKV, demonstrating an effective balance between memory efficiency and long-context performance.

Scalability to larger models. We further evaluate the scalability of *SCORE* on larger models, including Llama2-13B and Llama3-70B, under the same cache budget. As shown in Table 2, *SCORE* consistently achieves the best average performance on both models, improving over the strongest compressed baseline by $+1.84$ on Llama2-13B and $+0.59$ on Llama3-70B. Notably, *SCORE* maintains consistent gains across task groups, indicating that its benefits are not confined to a specific capability or benchmark subset. These results suggest that the layer–head diversity signals used by *SCORE* remain reliable across larger model scales.

Evaluation on LongBench V2. We further evaluate *SCORE* on LongBench V2 (Bai et al., 2024), a more challenging benchmark covering realistic ultra-long-context tasks. Table 3 reports the results on LLaMA3-8B-Instruct. Across both cache sizes (128 and 1024), *SCORE* consistently outperforms competitive baselines, showing that the method provides benefits regardless of memory budget. In the extreme compression setting with cache size 128, where most baselines struggle, *SCORE* achieves an average score of 26.04, surpassing HeadKV and CAKE by $+0.52$ and $+1.08$, respectively. The gains persist across difficulty levels and input lengths, with particularly clear advantages in the Hard and Medium scenarios where long-range reasoning is crucial. Notably, the improvement remains consistent when the cache budget increases from 128 to 1024, indicating that *SCORE* is not limited to extreme compression settings. Its advantage on longer inputs further suggests that redundancy-aware allocation remains effective as contextual dependencies become longer and more complex. This consistent behavior across cache budgets and difficulty levels further demonstrates the robustness of *SCORE* under diverse long-context evaluation settings. These results highlight that *SCORE* preserves critical information across different compression regimes and long-context conditions, supporting its effectiveness for practical long-context inference.



(a) Prefill stage latency breakdown.

(b) TTFT and decoding latency.

Figure 6: Latency analysis of *SCORE*. (a) Prefill stage latency breakdown on Mistral-7B and LLaMA2-7B across sequence lengths. (b) TTFT and decoding latency on Mistral-7B-Instruct-v0.3.

Alloc. Metric		Greedy	Single-Doc.	Multi-Doc.	Summ.	Few-shot	Avg.
$\mathcal{ID}$	$\mathcal{IN}$						
×	×	×	29.12	28.12	19.03	64.00	35.1
✓	×	×	30.23	29.15	19.78	65.00	36.0
×	✓	×	29.94	29.44	20.08	64.75	36.1
✓	✓	×	31.84	30.27	21.02	66.05	37.3
✓	✓	✓	**32.93**	**33.80**	**22.38**	**67.85**	**39.2**

Table 4: Component-wise ablation results on LongBench (Llama3-8B-Instruct, cache 128). Average performance over the four shown categories.

Method	Single-Doc.	Multi-Doc.	Summ.	Few-shot	Synthetic	Code
Full	**33.01**	33.65	**22.39**	66.92	**38.50**	**55.17**
Random	32.23	33.25	22.17	66.50	36.00	55.06
Top-k	31.10	29.11	20.74	65.20	36.50	53.39
Middle ($K = 200$)	32.93	33.80	22.38	**67.85**	37.25	55.28
Middle ($K = 400$)	32.91	**33.84**	22.21	67.51	38.00	55.11

Table 5: Ablation on sample selection strategies and sample size K for redundancy estimation on LongBench (Llama3-8B, cache size 128). Reported values are average scores per task category.

4.3 Efficiency Analysis

Computational Overhead Analysis. To assess the practical efficiency of *SCORE*, we analyze its additional operations during inference: (i) **distance matrix** computation, (ii) **penalty reweighting**, and (iii) **running-mean updates**. Figure 6a shows the normalized latency breakdown on Mistral-7B and LLaMA2-7B. The distance matrix—the dominant overhead—accounts for a diminishing fraction of inference latency as sequence length increases: from 2.15% at 4K to 0.36% at 32K on Mistral-7B, and from 2.45% at 0.5K to 0.45% at 4K on LLaMA2-7B. Since attention and FFN scale superlinearly, this overhead becomes marginal at long contexts. Overall, *SCORE* adds less than 1% latency where KV cache compression matters most, achieving a favorable accuracy-efficiency trade-off.

Evaluation on TTFT and Decoding Latency. We evaluate the practicality of *SCORE* in terms of time to first token (TTFT) and decoding latency. As shown in Figure 6b, *SCORE* achieves comparable latency to existing approaches across input lengths despite estimating redundancy across layers and heads. Rather than computing full pairwise distances, *SCORE* evaluates similarity only between adjacent layers using a controlled number of sampled vectors, substantially reducing computation. These results show that *SCORE* maintains comparable responsiveness while using a more expressive redundancy-aware scoring mechanism across layers and heads.

4.4 Ablation Study

Component-wise contribution. Table 4 presents a cumulative ablation of *SCORE* on Llama3-8B/LongBench with a cache size of 128. Since the proposed multi-level redundancy metrics ($\mathcal{ID}$ and $\mathcal{IN}$) drive hierarchical allocation, we evaluate metric-based allocation as a single component. Starting from uniform allocation with Top-k selection (35.1), hierarchical allocation improves the score to 37.3, yielding a $+2.2$ gain. Adding greedy token selection further increases the score to 39.2, providing an additional $+1.9$ gain and showing that diversity-aware selection mitigates head-level concentration. These results show that both metric-driven allocation and redundancy-aware selection are essential under strict cache budgets.

Sampling strategy & size. We analyze two design choices in redundancy estimation—*which* tokens to sample and *how many*—in Table 5 (Llama3-8B/LongBench, cache 128). Among Full, Random, Top-k, and Middle (ours), Middle closely matches Full while clearly outperforming Top-k and Random; Top-k underperforms because high-attention tokens are concentrated and fail to capture broader representational structure. Increasing K from 200 to 400 yields only marginal gains, so we adopt $K = 200$ as the default, keeping overhead negligible while matching larger sizes. This result indicates that middle-position sampling captures the redundancy structure without exhaustive attention observations, balancing estimation quality and overhead.

Method	Single-Doc.	Multi-Doc.	Summ.	Few-shot	Synthetic	Code	Avg.
FullKV	42.58	47.30	26.95	68.29	53.25	42.10	46.63
CAKE	35.18	44.19	18.77	56.63	53.25	36.34	40.22
HeadKV	35.84	42.17	19.58	65.06	51.80	36.41	41.52
SCORE	**37.28**	**44.27**	20.47	64.54	49.75	35.44	**41.88**
w/o $\mathcal{HU}$ (Alloc.)	37.24	43.24	20.24	62.89	49.00	36.00	41.30
w/o $\mathcal{HU}$ (Reweight.)	36.77	42.72	**20.77**	62.85	50.25	35.22	41.27
w/o $\mathcal{HU}$ (Both)	36.92	42.25	20.45	62.08	50.75	35.23	41.07
w/o Eq. (14)	36.21	43.52	19.74	**64.91**	**51.00**	**36.05**	41.70

Table 6: LongBench results and $\mathcal{HU}$ ablation on Qwen2.5-7B-Instruct-1M at cache budget 128. Ablation results are averaged over all 16 tasks.

Long-context evaluation and head uniqueness. We further evaluate *SCORE* on Qwen2.5-7B-Instruct-1M over all 16 LongBench tasks at a cache size of 128, with input lengths up to 31.5K tokens. This setting examines whether redundancy signals estimated from fixed samples remain effective as context length increases. As shown in Table 6, *SCORE* achieves the highest average among compression methods (41.88), outperforming HeadKV (41.52) and CAKE (40.22), demonstrating its effectiveness on substantially longer contexts. We further isolate head uniqueness ($\mathcal{HU}$), used in both head-budget allocation (Eq. (12)) and token reweighting (Eq. (14)). Removing $\mathcal{HU}$ entirely reduces the average from 41.88 to 41.07 (-0.81), while removing it only from allocation or reweighting results in drops of 0.58 and 0.61 points, respectively. The consistent degradation confirms that $\mathcal{HU}$ contributes to both hierarchical budget allocation and redundancy-aware token selection.

Consensus-token preservation. Reducing cross-head overlap may inadvertently suppress tokens that are repeatedly selected because they are genuinely important. We therefore examine sink and consensus tokens using labels derived from full-cache attention, independently of the eviction methods. Consensus tokens represent positions selected by more heads than expected by chance, while sink tokens capture attention-mass outliers. As shown in Table 7, *SCORE* preserves 100% of sink tokens and achieves the highest consensus-token retention (8.4%), while substantially reducing pairwise Jaccard overlap to 0.175 and increasing unique-token coverage to 0.438. Disabling the penalty in Eq. (14) increases overlap to 0.337 and reduces unique coverage to 0.191, while consensus retention also drops to 6.3%. These results indicate that the penalty primarily suppresses redundant cross-head selections rather than important shared tokens, improving contextual coverage without sacrificing critical consensus information across attention heads.

Method	Consensus ↑	Sink ↑	Jaccard ↓	Unique ↑	Avg. ↑
HeadKV	6.4%	99.6%	0.317	0.204	41.52
CAKE	5.2%	98.3%	0.499	0.303	40.22
SCORE	**8.4%**	**100.0%**	**0.175**	**0.438**	**41.88**
SCORE w/o Eq. (14)	6.3%	**100.0%**	0.337	0.191	41.70

Table 7: Consensus-token preservation and cross-head overlap at cache budget 128. Retention and overlap are measured on Llama-3-8B; LongBench averages are measured on Qwen2.5-7B-Instruct-1M.

Model	KL (Low / Rand. / High)	Spearman	Norm-matched
Llama-3-8B-Instruct	3.4 / 12.8 / 24.8	0.646	0.589
Mistral-7B-Instruct-v0.3	2.6 / 7.7 / 14.0	0.581	0.558
Qwen2.5-7B-Instruct	13.2 / 20.2 / 32.5	0.474	0.437
Qwen2.5-7B-Instruct-1M	12.4 / 18.0 / 30.8	0.505	0.458

Table 8: Head-swap validation using KL divergence and rank correlation across four models.

Functional validation of redundancy. To examine whether attention-distance reflects functional redundancy, we perform a head-swap intervention that replaces one head's output with that of another head, using 30 stratified head pairs per layer. As shown in Table 8, output KL consistently increases from low-distance to random and high-distance pairs across all four models, with stratified Spearman correlations of 0.474–0.646. This monotonic trend indicates that heads identified as similar by our attention-distance metric are also more safely substitutable at the model output. The correlations remain high after matching readout magnitudes, indicating that the relationship is not merely a magnitude artifact. These results support attention-distance as an effective and lightweight proxy for functional substitutability.

5 Conclusion

In this paper, we propose *SCORE*, a cache management framework that addresses structural redundancy in long-context processing. *SCORE* employs distance-based, multi-level similarity metrics to quantify redundancy across layers and attention heads, enabling hierarchical budget reallocation and redundancy-aware token selection. It explicitly models relational redundancy across layers and heads for cache budgeting. The additional analyses further validate that its redundancy signals capture functional substitutability while preserving important consensus information across heads. Extensive experiments on long-context benchmarks demonstrate that *SCORE* consistently outperforms existing methods under tight cache constraints, particularly on complex reasoning and retrieval tasks.

Limitations

Despite its effectiveness in managing multi-level redundancy in the KV cache, *SCORE* has several limitations. (1) It requires sampling representations from each attention head and computing pairwise distances, introducing additional prefill-time computation compared with simple statistical metrics (*e.g.*, variance or entropy). Although our cascading design and bounded sampling keep this overhead small in our evaluations, further optimization may be required for broader serving configurations. (2) *SCORE* relies on a fixed cosine-distance metric. While our head-swap analysis validates its correlation with functional substitutability across multiple models, a fixed metric may not capture all task- or context-dependent redundancy patterns. Task-aware or learned similarity measures remain an interesting direction. (3) This work focuses on inference-time cache management; extending *SCORE* to training scenarios such as fine-tuning or continual learning remains an open direction for future research. (4) *SCORE* focuses on eviction-based KV cache compression and does not jointly optimize orthogonal compression dimensions such as KV cache quantization or low-rank approximation. Since quantization changes numerical precision and low-rank methods modify representation rank, these techniques are complementary to *SCORE*'s redundancy-aware allocation and token selection. Combining *SCORE* with such methods could further improve the memory–accuracy trade-off, which we leave for future work.

Acknowledgments

This work was partly supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program) (RS-2025-02307330, Development and Validation of On-Device AI Semiconductor for Manufacturing Automation Robots through sVLM-Based Situational Awareness), funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea), and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIP) (RS-2025-02214652, Development of SoC Technology for AI Semiconductor-Converged Pooled Storage/Memory).

References

Shantanu Acharya, Fei Jia, and Boris Ginsburg. 2024. Star attention: Efficient llm inference over long sequences. *arXiv preprint arXiv:2411.17116*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1:1.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2024. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and 1 others. 2024. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. *arXiv preprint arXiv:2004.04010*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International conference on machine learning*, pages 2793–2803. PMLR.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2024. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. *arXiv preprint arXiv:2410.19258*.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Mo Li, Songyang Zhang, Yunxin Liu, and Kai Chen. 2024a. Needlebench: Can llms do retrieval and reasoning in 1 million context window? *arXiv preprint arXiv:2407.11963*.

Xiang Li, Zhenyan Lu, Dongqi Cai, Xiao Ma, and Mengwei Xu. 2024b. Large language models on mobile devices: Measurements, analysis, and insights. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, pages 1–6.

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

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

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

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

Siyu Ren and Kenny Q Zhu. 2024. On the efficacy of eviction policy for key-value constrained generative language model inference. *arXiv preprint arXiv:2402.06262*.

Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. 2024. Keep the cost down: A review on methods to optimize llm's kv-cache consumption. *arXiv preprint arXiv:2407.18003*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. 2024. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*.

An Yang, Baosong 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, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2022. Are all layers created equal? *Journal of Machine Learning Research*, 23(67):1–28.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and 1 others. 2024. infty bench: Extending long context evaluation beyond 100k tokens. *arXiv preprint arXiv:2402.13718*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

A Detailed Experimental Setup

Baseline Models. We evaluate *SCORE* on representative open-source LLMs spanning a wide range of context lengths (from 4K to 128K tokens) and attention architectures. To verify generalization across both Multi-Head Attention (MHA) and Grouped-Query Attention (GQA), we include: **(i)** Llama2-7B (Touvron et al., 2023) and Llama2-13B (Touvron et al., 2023) as MHA models; and **(ii)** Llama3-8B-Instruct (Grattafiori et al., 2024), Llama3-70B-Instruct (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Qwen2.5-7B-Instruct (Yang et al., 2024) as GQA models. This selection covers four model families and parameter scales from 7B to 70B, allowing us to assess *SCORE* under diverse architectural and capacity regimes.

Baselines. We compare *SCORE* against representative KV cache compression methods covering both layer-level and head-level allocation strategies: StreamingLLM (Xiao et al., 2023), H2O (Zhang et al., 2023), SnapKV (Li et al., 2024c), PyramidKV (Cai et al., 2024), HeadKV (Fu et al., 2024), and CAKE (Qin et al., 2025). We additionally report Full KV (no compression) as an upper-bound reference. To ensure a fair comparison, all baselines adopt the same dynamic reallocation scheme during prefilling, which adjusts cache usage to retain informative tokens under constrained budgets.

Tasks and Benchmarks. We evaluate model performance under compressed KV cache settings using five complementary long-context benchmarks:

- **LongBench** (Bai et al., 2023): A multi-task benchmark covering 16 datasets across six categories—single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion.
- **LongBench V2** (Bai et al., 2024): An extended version designed to overcome limitations of prior datasets by incorporating more realistic and challenging scenarios across varying difficulty levels (Easy/Hard) and context lengths (Short/Medium/Long).
- **NeedleBench** (Li et al., 2024a): A retrieval-focused benchmark that evaluates fine-grained information retrieval under varying context lengths and depths.
- **Reasoning-in-a-Haystack** (Kuratov et al., 2024): A reasoning-focused benchmark designed to measure long-range multi-step reasoning ability.
- **InfiniteBench** (Zhang et al., 2024): An ultra-long-context benchmark with an average sequence length of 145K tokens (max 214K), spanning five domains—Retrieval, Code, Math, Novels, and Dialogue—designed to stress-test long-context understanding and reasoning beyond 100K tokens.

Together, these benchmarks span the major dimensions of long-context evaluation, including retrieval, reasoning, summarization, and in-context learning.

Cache Budget Settings. We evaluate all methods across a wide range of KV cache budgets, from severely constrained settings (64L) to near-full-cache settings (1024L), where L denotes the number of layers. This range allows us to characterize the accuracy-efficiency trade-off across different operational regimes, with particular emphasis on low-cache scenarios ($B_{\text{total}} \leq 128L$) where the gap between methods is most pronounced.

Implementation. All experiments were conducted on a single NVIDIA H100 80GB GPU with FP16 precision. To ensure fair comparison, all methods were evaluated under identical conditions, including the same prompt formatting, decoding strategy (greedy decoding), and cache capacity. We reuse the official implementations of baseline methods whenever available and re-tune their hyperparameters following the protocols described in the original papers.

B More Implementation Settings

Details of Hyper-parameters. The algorithm of *SCORE* consists of three main stages: multi-level redundancy estimation, layer-wise budget allocation and cache management, and salient token selection. In the redundancy estimation stage (Section ***3.2 Multi-Level Redundancy Scoring***), we sample representations from each attention head and compute pairwise distances to extract multi-level redundancy. This process captures similarity patterns across layers, serving as the basis for quantifying cache necessity. In particular, we compute the $\mathcal{IN}$ between consecutive layers to assess the novelty of information introduced at each layer. To stabilize estimation, we update the accumulated baseline using a running mean, as defined in Eq. (7). In the budget allocation and cache manage-

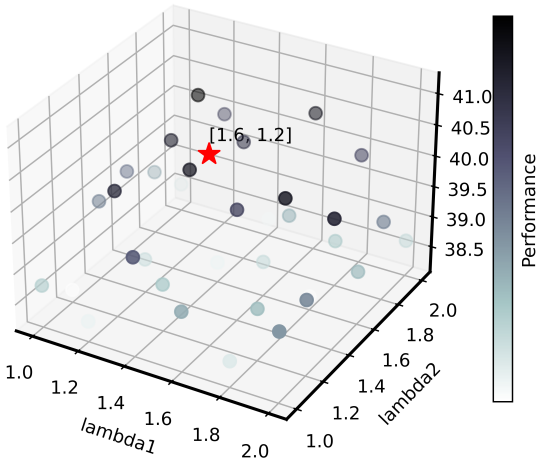


Figure 7: Performance comparison on LongBench for different λ_1 and λ_2 combinations.

ment stage (Section ***3.3 Hierarchical Budget Allocation***), we assign layer-wise KV cache budgets based on the estimated redundancy and perform eviction accordingly. We perform a grid search over $\lambda_1, \lambda_2 \in [1.0, 2.0]$ with a 0.2 step size for adaptive budget scaling, and use the same default setting across all main experiments. See Figure 7 for performance across different combinations. In the final stage (Section ***3.4 Greedy Token Selection***), no additional hyperparameter is introduced; token selection is performed using the allocated head-wise budgets and the penalized attention scores.

Ablation on Sample Size. To keep the computation cost stable regardless of sequence length, we fix the number of sampled tokens K when estimating head-wise similarity. Nevertheless, in extremely long-context scenarios, model performance could in principle depend on the choice of K. Table 9, therefore, presents an ablation on Llama3-8B-Instruct with LongBench V2 (input sequences extended up to $\sim$2M tokens) under a cache size of 128, comparing SCORE with $K = 200$ and $K = 400$ against baseline methods. The results show that the performance gap between different K values remains minimal, confirming that even a relatively small K is sufficient and that SCORE is robust to the sampling.

C Reasoning and Redundancy Analysis

Reasoning-in-a-Haystack Results. To further assess the generalizability of *SCORE*'s long-context reasoning, we conduct experiments on diverse benchmarks. Table 10 presents results on the Reasoning in a Haystack task under limited cache budgets and varying input lengths. *SCORE* consistently outperforms baselines across all lengths, indicating its ability to preserve precise reasoning. Notably, it exhibits stable performance degradation at longer contexts (16k–32k) and achieves the best average score, suggesting robust retention of critical reasoning signals under severe cache constraints. This demonstrates that *SCORE*'s cache strategy—balancing importance and diversity—is effective for long-range dependency reasoning.

Method	Easy	Hard	Short	Medium	Long	Avg.
Full KV	31.25	25.08	34.44	24.19	22.22	27.44
Llama3-8B-Instruct, Cache size=128						
CAKE	29.27	21.13	**32.98**	21.22	20.18	24.96
HeadKV	29.53	22.74	32.25	22.97	20.10	25.52
SCORE ($K = 200$)	**29.99**	23.22	32.87	23.75	20.35	26.04
SCORE ($K = 400$)	29.85	**23.40**	32.71	**23.80**	**20.61**	**26.07**

Table 9: Ablation study on Llama3-8B-Instruct with LongBench V2. We analyze the effect of different sample sizes ($K = 200$ and $K = 400$) for our method (SCORE) under cache size 128, compared against CAKE and HeadKV.

Method	0k	1k	2k	4k	8k	16k	32k	Avg.
FullKV	61.30	55.30	53.40	42.10	40.30	34.00	31.80	45.46
SnapKV	55.40	50.20	46.40	37.20	35.00	32.80	29.20	40.89
PyramidKV	57.20	50.80	47.60	36.20	36.20	31.40	28.20	41.09
HeadKV	**58.60**	53.80	**52.20**	38.20	37.60	31.80	30.40	43.23
CAKE	58.40	54.00	51.30	**38.40**	37.20	31.80	30.20	43.04
SCORE	58.40	**54.20**	51.80	38.30	**37.80**	**32.10**	**30.60**	**43.31**

Table 10: Reasoning-in-a-Haystack results on Mistral-7B-Instruct with 128 KV cache. Scores are averaged over QA1–QA5 tasks at each context length.

Token-Level Redundancy. Across layers, we observe that multiple heads frequently select overlapping tokens, even when their budgets differ. This concentration leads to redundant evidence being repeatedly gathered from the same positions, limiting overall coverage and reducing the ability to capture diverse contextual cues. As illustrated in Figure 8, several heads consistently attend to the same high-saliency tokens and select them, producing narrow regions of activation that dominate the token-selection process. This redundancy becomes more pronounced under smaller cache budgets, where heads tend to collapse onto a shared subset of tokens rather than distributing attention across complementary parts of the sequence. These patterns motivate a redundancy-aware selection strategy that mitigates token-level overlap and promotes complementary evidence across heads.

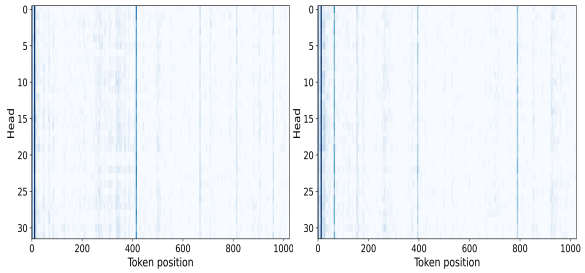
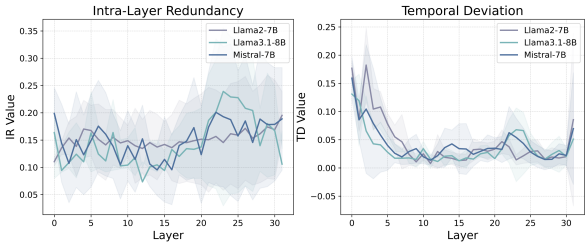
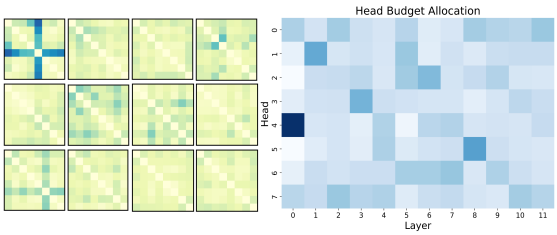


Figure 8: Aggregated head-wise token selection frequency across all layers. Evaluated on NarrativeQA (left) and 2WikiMQA (right).



(a) Layer-level budget



(b) Head-level budget

Figure 9: Redundancy metrics used for budget allocation. (a) Layer-wise distributions of $\mathcal{ID}$ and $\mathcal{IN}$ averaged over LongBench. (b) Head-wise redundancy for the first 12 layers on NarrativeQA and corresponding budget allocation.

C.1 Justifying Redundancy-aware Modeling

Redundancy-Aware Allocation. *SCORE* adaptively allocates cache budgets across layers and heads based on quantified redundancy levels. As illustrated in Figure 9, components exhibiting greater information diversity tend to receive larger cache budgets, while highly redundant ones are assigned less. For instance, lower layers tend to exhibit lower similarity to others, resulting in high $\mathcal{IN}$ in Figure 9(a), and are thus prioritized in layer-level budget assignment, receiving relatively higher budgets in practice. Figure 9(b) further reveals that certain heads within these layers carry unique information, leading to increased head-level budget allocations. This analysis suggests that *SCORE* effectively leverages distance-based similarity metrics to identify structural redundancy and maximize informational diversity under constrained resources.

D Additional Detailed Analysis and Visualization

D.1 Effect of Distance Metric Choice

Figure 10 visualizes pairwise similarities across all heads and layers using various distance metrics: Cosine similarity (COS), Pearson correlation (COR), Jensen–Shannon distance (JSD), Bhattacharyya coefficient (BCD), and Euclidean distance (ECD). Bright regions indicate high similarity, with all metrics revealing consistent alignment across specific layers and heads. These metrics span a diverse range of similarity formulations, from distribution-based measures (JSD, BCD) to correlation- or geometry-based ones (COR, ECD), providing a comprehensive comparison. Despite differing statistical bases, these metrics capture similar redundancy signals, suggesting pronounced structural redundancy in model representations. This observation indicates that redundancy is not an artifact of a particular distance metric, but rather a consistent structural property across attention heads and layers. Therefore, SCORE can exploit these stable redundancy patterns for budget allocation and token selection. We adopt cosine distance as the default metric due to its strong empirical performance and computational simplicity.

D.2 Visualization of Distance Matrix

To further analyze the behavior of the cosine distance metric in our retrieval framework, we visualize the pairwise distance matrices computed using cosine similarity across different datasets in LongBench, as shown in Figure 11. Each matrix represents the inter-token similarity structure within heads, with darker regions indicating lower similarity (*i.e.*, higher cosine distance). Across datasets, we observe consistent patterns of redundancy, where certain groups of heads exhibit strong mutual similarity. While the specific patterns vary depending on the dataset domain and structure, the presence of high-similarity clusters is a common characteristic. These clusters often correspond to repeated representations. Such redundancy can degrade retrieval efficiency and content diversity if not properly managed. *SCORE* addresses this by leveraging cosine distance not only to capture salient content but also to suppress over-represented or semantically repetitive tokens. This behavior is especially beneficial in budget-constrained settings, where the selection of maximally informative yet diverse tokens is critical.

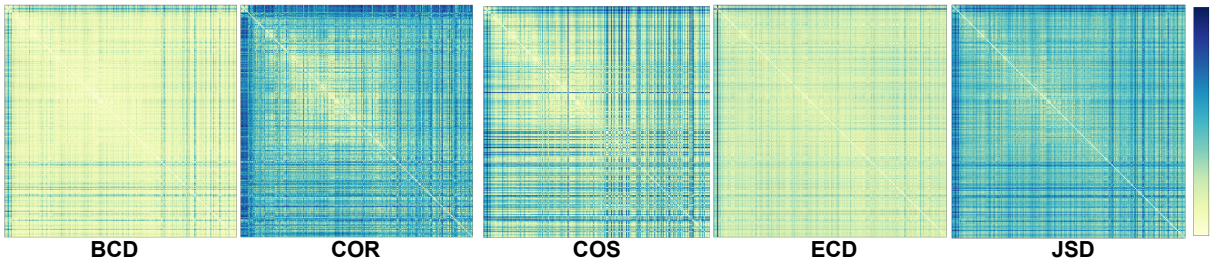


Figure 10: Head-to-head similarity matrices across various metrics for NarrativeQA. Each value represents the average distance between 200 sampled attention vectors per head.

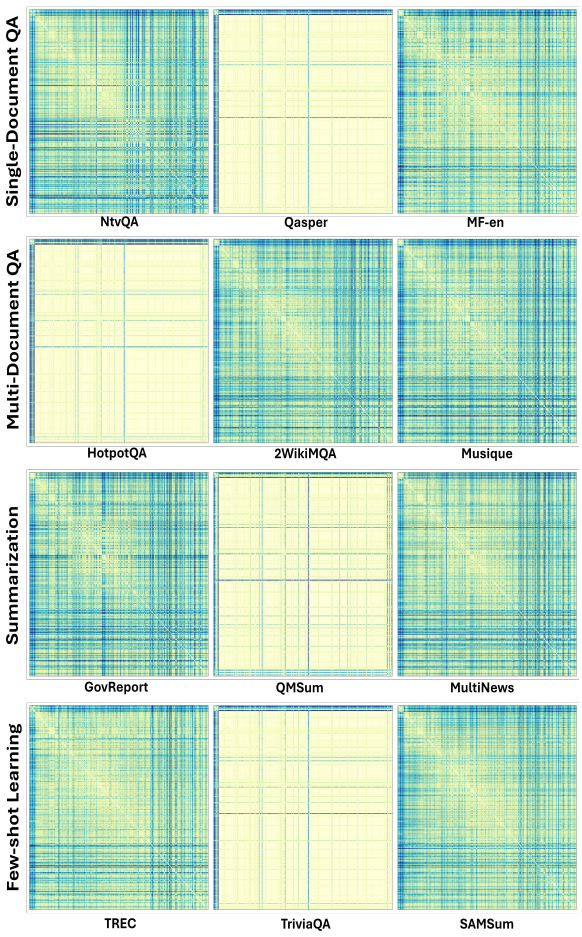


Figure 11: Head-wise cosine distance matrices for various datasets in LongBench.

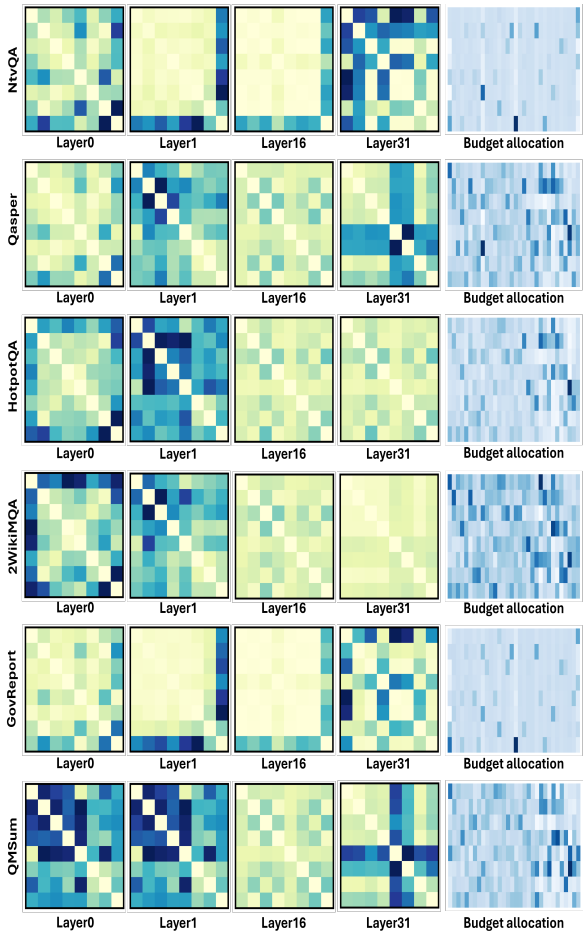


Figure 12: Layer-wise distance matrices and head-level budget allocation across LongBench datasets.

D.3 Visualization of Budget Allocation

To better understand how retrieval budgets are distributed across the model's architecture, we visualize intra-layer similarity patterns across different datasets in LongBench, as shown in Figure 12. Each heatmap captures pairwise similarity within heads at each layer, providing insight into redundancy and representational diversity. We observe a consistent trend across datasets: the lower layers generally exhibit higher diversity, as indicated by lower intra-head similarity. These layers tend to capture localized, fine-grained features, making their token representations less redundant. Consequently, they receive a larger share of the retrieval budget, allowing more tokens to be selected from them. In contrast, middle layers often show pronounced redundancy, with many heads producing highly similar token embeddings. The final column in Figure 12 shows the head-level budget distribution, where heads with high distinctiveness are allocated more budget. This fine-grained allocation ensures that informative and non-redundant heads are prioritized, aligning with the principles in Section ***3.3 Hierarchical Budget Allocation***.

Despite variations in domain and task, several datasets exhibit notably similar similarity profiles across layers and heads, suggesting that token representation patterns are influenced not only by data but also by the model's inherent architecture. This structural consistency points to a promising direction for developing more generalized, architecture-aware retrieval strategies. Importantly, such consistency does not imply identical redundancy distributions across all settings, but rather indicates stable relative trends that can be reliably exploited for budget allocation. By grounding retrieval decisions in these structural patterns, the proposed approach avoids reliance on dataset-specific heuristics and remains applicable across diverse evaluation scenarios. This observation further supports the use of redundancy-aware, hierarchical allocation as a principled design choice.

E Effect of the distance metric

We study how the choice of distance metric for constructing the pairwise distance matrix affects LongBench performance under a fixed cache budget (128) on Qwen2.5-7B-Instruct. We compare JSD, ECD, COR, BCD, and COS. As shown in Table 11, COS distance yields the strongest overall performance among the tested metrics, consistently preserving accuracy across diverse LongBench categories. In particular, COS achieves the best scores on Single-Doc. QA, Multi-Doc. QA, Few-shot, and Synthetic tasks, while remaining competitive on Summarization and Code. This suggests that cosine distance provides a stable similarity signal for identifying redundant heads and tokens across heterogeneous task types. Figure 13 presents a distance metric ablation across different cache budgets. On both Llama3-8B-Instruct and Mistral-7B-Instruct-v0.3, COS distance consistently achieves the best average scores and shows the smallest performance degradation under tight cache budgets. Given its robustness, we adopt COS distance as the default distance metric and use it in all experiments.

F SCORE Algorithm Overview

Algorithm 1 summarizes the end-to-end pipeline of SCORE, which corresponds to the overall framework described in Sections 3.1–3.4 of the main paper. Under a total KV cache budget B_{total}, the algorithm implements multi-level redundancy scoring, hierarchical budget allocation, and redundancy-aware token selection in a unified manner. Specifically, for each cascading stage $m \in \{0, \ldots, L-1\}$, we update the cascading cache set $C^{(m)}$ and compute a distance matrix S. From this matrix, we derive the stage-wise intra-layer diversity $\mathcal{ID}_m$ and inter-layer novelty $\mathcal{IN}_m$. By accumulating these statistics up to stage m, SCORE determines the layer-wise cache budgets $B^{(m)} = \{B_l^{(m)} \mid l \in [0, m]\}$, reflecting the relative diversity and novelty of each layer. Next, given the attention matrix $A_m \in \mathbb{R}^{H \times S}$, we estimate the head-wise importance $M_{m,h}$ and the head uniqueness score $\mathcal{HU}_{m,h}$. These two quantities are used to allocate the head-wise budgets $B_{m,h}^{(m)}$ within each layer. Finally, SCORE performs redundancy-aware token selection by computing, for each (head, token) pair, the dependency score $U_{m,h,s}$ that measures how strongly head h depends on token s relative to all heads in the layer. Combining $U_{m,h,s}$ with the head uniqueness score $\mathcal{HU}_{m,h}$ through a multiplicative penalty, we form penalized attention scores $\widetilde{A}_m$ and apply $I_{m,h}^{(m)} \leftarrow \text{TopK}(\widetilde{A}_m[h], B_{m,h}^{(m)})$ for each head, retaining only the selected $\hat{K}_{m,h}^{(m)}$ and $\hat{V}_{m,h}^{(m)}$ in $C^{(m)}$. The final TopK step therefore preserves high-scoring tokens while reducing redundant selections across heads. Overall, Algorithm 1 operationalizes the SCORE framework by jointly reallocating cache budgets across layers and heads and selectively preserving salient yet non-redundant tokens, enabling stable and effective KV cache eviction under strict memory budgets.

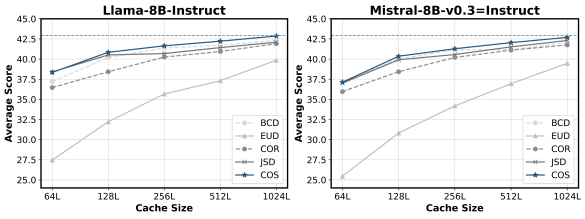


Figure 13: LongBench performance by cache budget for each distance matrix.

Table 11: Performance comparison on Qwen2.5-7B-Instruct over LongBench under various distance metrics. Results are measured with a cache size of 128.

Method	Single-Doc. QA	Multi-Doc. QA	Summ.	Few-shot	Synthetic	Code
Full KV	41.14	44.09	25.30	69.05	51.23	63.61
Qwen2.5-7B-Instruct, Cache size=128						
JSD	32.67	37.26	22.45	65.90	47.14	55.74
ECD	30.92	35.41	22.36	63.18	42.27	39.48
COR	34.10	38.81	23.38	66.00	46.30	48.19
BCD	32.74	37.53	22.71	65.50	45.34	46.22
COS	34.99	40.84	23.00	66.20	51.04	55.43

Algorithm 1 Multi-level budget allocation

Input: Accumulated attention scores $\{A_m \in \mathbb{R}^{H\times S}\}_{m=0}^{L-1}$, number of layers L, number of heads H, total cache budget B_{total}

Output: Retained cache set $C = \{C^{(m)}\}_{m=0}^{L-1}$

// Layer-wise budget allocation with cascading cache updates

for stage $m = 0$ to $L-1$ **do**
  $C^{(m)} \leftarrow C^{(m-1)} \cup \{K^{(m)}, V^{(m)}\}$
  Compute distance matrix $S \in \mathbb{R}^{H\times 2H}$ using Eq. (4)
  Compute $\mathcal{ID}_m$ and $\mathcal{IN}_m$ from S using Eqs. (5)–(7)
  Update the running statistics $\mathcal{ID}^{(m)}$ and $\mathcal{IN}^{(m)}$
  Compute layer priority scores $\mathcal{N}_l$ using Eq. (9)
  Allocate layer-wise budgets $B^{(m)} = \{B_l^{(m)} \mid l \in [0, m]\}$ using Eq. (10)
  // Head-wise budget allocation
  Compute head-wise importance $M_{m,h}$ from A_m using Eq. (11)
  Compute head uniqueness $\mathcal{HU}_{m,h}$ for all heads using Eq. (8)
  Allocate head-wise budgets $B_{m,h}^{(m)}$ using $M_{m,h}$ and $\mathcal{HU}_{m,h}$ as in Eq. (12)
  // Redundancy-aware token selection
  Compute per-(head, token) dependency $U_{m,h,s}$ using Eq. (13)
  Compute penalized attention scores $\widetilde{A}_m$ using Eq. (14)
  for head $h = 0$ to $H-1$ **do**
    $I_{m,h} \leftarrow \text{TopK}\big(\widetilde{A}_m[h],\, B_{m,h}^{(m)}\big)$
    $\hat{K}^{(m)}[h],\, \hat{V}^{(m)}[h] \leftarrow K^{(m)}[I_{m,h}],\, V^{(m)}[I_{m,h}]$
    $C^{(m)}[h] \leftarrow \{\hat{K}^{(m)}[h],\, \hat{V}^{(m)}[h]\}$
  end for
end for
return $C = \{C^{(m)}\}_{m=0}^{L-1}$

G Experiments on InfiniteBench Dataset

In this section, we evaluate the effectiveness of the proposed method, *SCORE*, on InfiniteBench—a challenging benchmark specifically designed to assess the long-context processing capabilities of large language models. A detailed comparison of results across models and cache budgets is provided in Tables 12 and 13. Compared to prior benchmarks such as LongBench, InfiniteBench introduces substantially greater difficulty by incorporating ultra-long input sequences, with an average context length of 145K tokens and a maximum of up to 214K tokens. The benchmark spans five diverse domains—Retrieval, Code, Math, Novels, and Dialogue—providing a comprehensive testbed for evaluating a model's ability to understand, reason over, and extract salient information from extremely long contexts. We conduct experiments using both Llama3-8B-Instruct and Mistral-7B-Instruct-v0.3, evaluating performance across 10 datasets from InfiniteBench under two KV cache budgets: 128L and 1024L. At a cache size of 128L, *SCORE* achieves the highest average accuracy of 18.00 on Mistral-7B-Instruct-v0.3, outperforming the previous best-performing method, CAKE, which attains 17.57—a margin of 0.42. When the cache size is increased to 1024L, *SCORE* further improves to 19.67, again surpassing CAKE (18.99), with a larger margin of 0.68. These results demonstrate the effectiveness of *SCORE* in compressing and retaining salient information within memory-constrained settings, even under extreme sequence lengths. Importantly, *SCORE* maintains consistent performance improvements across both low and high cache budgets, underscoring its robustness and scalability. This highlights *SCORE* as a compelling solution for long-context language modeling, effectively balancing memory efficiency with task performance across a wide range of domains and input lengths.

H Experiments on LongBench Dataset

We provide a full breakdown of the LongBench evaluation results for Llama-2-7B, Llama-2-13B, Llama-3-8B-Instruct, and Mistral-7B-v0.3-Instruct. The results are presented in ascending order of cache size: 64L (Table 14), 256L (Table 15), 512L (Table 16).

Results on Llama3-8B-Instruct. *SCORE* consistently outperforms prior works across all cache sizes on the LongBench benchmark. The advantage is especially prominent in the low-cache regime, notably at 64L (Table 14), where *SCORE* demonstrates a significantly higher score compared to other methods. A key observation is that existing works often select redundant key-value pairs under tight cache budgets, leading to poor coverage and lower performance. In contrast, SCORE maintains accuracy even at 64L through diversity-aware selection, yielding more informative memory allocation.

Results on Mistral-7B-Instruct-v0.3. Similar to the observations with Llama3-8B-Instruct, our method (SCORE) shows strong and stable performance across cache sizes when evaluated on Mistral-7B-Instruct-v0.3. Notably, even under the extreme low-cache setting of 64L, *SCORE* preserves 77.1% of its performance relative to the full-cache setting, demonstrating robustness. Existing methods often select redundant key-value pairs under tight budgets, whereas SCORE maintains competitive accuracy at 64L through complementary memory allocation.

	Method	Retrieval		Code		Math	Novels				Dialogue
		Ret.PassKey	Ret.Number	Code.Debug	Code.Run	Math.Find	En.Sum	En.QA	En.choice	Chn.QA	En.Dia
Llama3-8B -Instruct	StreamingLLM	4.98	4.46	39.00	0.50	19.64	19.49	0.01	41.06	1.94	0.00
	H2O	5.30	5.57	40.19	0.50	20.32	20.01	0.01	41.16	2.21	0.00
	HeadKV	6.14	6.14	43.54	1.50	26.51	23.37	0.09	47.43	3.67	0.00
	SnapKV	6.34	6.32	43.32	1.50	25.83	23.45	0.10	48.25	3.41	0.00
	PyramidKV	6.48	6.21	43.44	1.50	26.24	23.32	**0.16**	48.03	3.16	**0.50**
	CAKE	6.22	**6.51**	43.48	1.50	26.74	**25.11**	0.05	48.16	3.75	**0.50**
	SCORE(ours)	**6.63**	5.83	**43.65**	**2.00**	**26.91**	23.71	0.03	**48.34**	**3.99**	**0.50**
Mistral-7B -Instruction-v0.3	StreamingLLM	24.10	5.56	28.17	0.00	21.56	20.03	0.10	40.56	8.01	0.00
	H2O	25.02	5.63	29.28	0.25	22.74	20.12	0.10	40.98	7.89	0.00
	HeadKV	26.75	6.27	32.74	0.25	27.43	22.74	0.31	49.10	**10.32**	0.50
	SnapKV	25.95	6.95	31.47	**0.50**	22.57	**23.14**	0.32	48.76	9.69	0.00
	PyramidKV	26.78	6.69	**33.25**	0.25	23.43	21.92	0.28	49.53	9.01	0.00
	CAKE	26.68	6.64	32.49	**0.50**	28.14	21.45	0.33	49.56	9.93	0.00
	SCORE(ours)	**26.91**	**7.03**	32.95	**0.50**	**28.64**	22.25	**0.39**	**50.14**	10.15	**1.00**

Table 12: Performance comparison over 10 datasets of InfiniteBench on Llama3-8B-Instruct and Mistral-7B-Instruct-v0.3. Results are measured with 128L cache. The best result is highlighted in **bold**, the second best in underline.

	Method	Retrieval		Code		Math	Novels				Dialogue
		Ret.PassKey	Ret.Number	Code.Debug	Code.Run	Math.Find	En.Sum	En.QA	En.choice	Chn.QA	En.Dia
Llama3-8B -Instruct	StreamingLLM	5.89	5.78	40.78	1.50	25.21	20.79	0.05	41.87	2.78	0.50
	H2O	5.48	6.14	42.23	1.00	25.49	21.31	0.05	42.65	3.12	0.50
	HeadKV	6.73	6.61	44.53	2.00	26.98	25.10	0.04	49.87	3.77	0.50
	SnapKV	6.42	6.58	44.91	2.50	27.13	24.64	0.05	48.42	3.34	0.50
	PyramidKV	6.59	6.44	44.87	2.25	27.17	25.00	0.15	49.47	4.10	**1.00**
	CAKE	6.71	6.76	44.79	2.50	27.43	24.28	0.10	50.12	3.68	0.50
	SCORE(ours)	**6.83**	**6.81**	**45.21**	**3.00**	**27.45**	**25.15**	**0.25**	**50.71**	**4.15**	**1.00**
Mistral-7B -Instruction-v0.3	StreamingLLM	26.00	8.51	30.28	0.50	25.67	21.01	0.05	47.13	8.78	0.00
	H2O	26.21	8.98	29.78	0.50	26.97	20.31	0.09	48.78	9.24	0.00
	HeadKV	27.00	10.12	35.28	1.00	28.71	21.65	**0.45**	50.51	10.48	0.00
	SnapKV	26.98	9.37	33.25	1.00	30.29	21.68	0.19	49.79	10.59	1.00
	PyramidKV	26.45	9.49	34.26	1.00	26.86	22.24	0.33	49.65	10.54	1.00
	CAKE	27.02	12.15	36.80	1.00	29.71	22.27	0.33	49.23	10.34	1.00
	SCORE(ours)	**27.13**	**12.25**	**39.15**	**1.50**	**30.73**	**22.73**	0.28	**50.73**	**10.68**	**1.50**

Table 13: Performance comparison over 10 datasets of InfiniteBench on Llama3-8B-Instruct and Mistral-7B-Instruct-v0.3. Results are measured with 1024L cache. The best result is highlighted in **bold**, the second best in underline.

	Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code	
		NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P
Llama2-7B	StreamingLLM	10.70	15.36	27.88	24.20	20.35	5.89	14.85	16.23	18.40	55.00	78.85	34.75	2.50	1.50	50.25	44.65
	H2O	11.33	17.26	29.21	24.49	21.46	7.84	16.49	18.00	20.00	62.00	80.30	36.29	5.00	2.50	51.46	46.19
	HeadKV	14.41	18.80	27.95	26.94	23.85	7.93	17.84	19.56	20.65	**63.00**	81.54	37.47	5.50	4.00	53.09	47.89
	SnapKV	13.67	18.05	26.80	28.00	21.38	7.49	16.78	19.59	19.85	57.50	81.40	36.98	5.50	3.50	51.60	45.96
	PyramidKV	14.75	18.25	28.57	29.27	23.12	7.50	17.42	19.56	19.87	60.00	82.14	36.26	5.50	3.50	52.13	46.23
	CAKE	15.49	**18.88**	31.97	**29.33**	**24.65**	9.35	**18.62**	19.88	**22.21**	**63.00**	82.82	**38.62**	**6.00**	**6.00**	**54.46**	**48.53**
	SCORE(ours)	**15.61**	18.80	**32.15**	29.21	23.98	**10.14**	18.51	**20.05**	20.67	**63.00**	**83.49**	37.49	**6.00**	5.50	53.88	47.71
Llama2-13B	StreamingLLM	10.45	7.56	21.95	9.03	7.13	1.61	17.45	16.75	17.73	60.00	82.10	37.31	0.50	12.50	32.45	35.89
	H2O	11.84	9.31	23.36	9.12	9.00	3.26	18.88	19.39	19.93	59.50	85.94	37.86	1.50	14.72	37.67	36.07
	HeadKV	11.99	12.14	23.53	10.65	9.03	3.70	19.28	19.86	19.85	64.00	**84.74**	38.48	**2.50**	15.75	35.92	36.22
	SnapKV	11.51	10.87	22.92	10.24	8.84	3.20	18.60	19.25	19.19	50.50	86.84	36.28	2.00	**16.75**	34.97	33.60
	PyramidKV	12.55	11.44	**24.53**	**10.84**	**10.04**	**4.01**	18.92	19.83	20.15	57.00	86.31	38.18	2.00	16.25	37.19	35.49
	CAKE	12.69	10.11	22.41	10.21	9.70	2.81	20.03	20.20	21.63	67.50	87.12	39.29	2.00	15.75	42.76	**41.25**
	SCORE(ours)	**12.99**	**12.53**	24.34	10.61	9.91	3.70	**20.45**	**20.31**	**22.21**	**68.00**	87.50	**40.15**	2.00	**16.75**	**43.31**	41.06
Llama3-8B -Instruct	StreamingLLM	20.26	16.29	30.02	34.81	25.73	13.70	15.26	19.30	12.94	60.50	80.77	30.62	4.63	60.50	48.87	43.84
	H2O	20.76	17.78	32.43	38.28	27.08	16.01	17.72	20.61	16.36	61.50	88.26	35.95	4.88	68.00	50.56	48.41
	HeadKV	**23.67**	17.46	32.72	39.81	27.50	17.39	19.14	22.12	20.04	65.00	90.20	37.27	4.77	69.00	54.68	**51.52**
	SnapKV	21.56	17.98	32.35	38.45	26.40	17.01	18.38	21.99	18.86	51.50	89.32	36.11	5.00	69.50	53.13	49.81
	PyramidKV	21.76	18.56	33.01	39.45	28.47	17.42	18.58	21.77	18.58	58.50	88.14	37.26	5.00	69.50	52.05	47.27
	CAKE	22.06	18.17	32.98	40.53	31.56	**17.98**	19.94	**22.30**	21.27	70.50	90.40	**38.85**	**6.00**	69.50	54.62	49.75
	SCORE(ours)	22.97	**18.81**	**33.34**	**41.11**	**31.63**	17.94	**20.43**	22.20	**21.61**	**72.00**	**91.00**	38.81	5.50	**70.00**	**56.13**	50.91
Mistral-7B -Instruction-v0.3	StreamingLLM	19.54	25.07	37.66	40.13	21.43	12.69	17.47	16.93	15.99	51.00	84.00	37.20	4.00	70.00	41.44	39.34
	H2O	20.10	25.40	37.92	40.48	21.78	13.00	17.73	17.51	16.37	57.00	84.50	37.62	4.50	75.00	41.83	39.96
	HeadKV	23.18	26.31	38.77	40.94	21.20	13.38	18.51	17.68	16.74	62.50	85.00	38.47	4.50	69.50	42.35	41.38
	SnapKV	22.21	26.53	37.83	39.96	20.10	13.00	17.20	17.33	16.00	50.00	85.00	36.54	5.00	76.50	41.22	38.60
	PyramidKV	21.89	26.53	37.90	40.52	22.17	13.41	17.78	17.69	**19.66**	54.50	84.00	36.56	**5.50**	77.50	40.16	37.42
	CAKE	**23.89**	26.89	**39.12**	42.08	25.26	**13.97**	**19.38**	18.03	18.25	64.50	85.00	40.22	5.00	78.00	45.02	42.94
	SCORE(ours)	23.87	**27.92**	38.99	**42.15**	**25.33**	13.93	19.22	**18.99**	18.91	**67.00**	**85.50**	**40.57**	5.00	**78.50**	**45.23**	**44.15**

Table 14: Performance comparison over 16 datasets from LongBench. Results are measured with a cache size of 64. The best score is marked in **bold**, and the second best is marked with underline.

		Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code	
	Method	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P
Llama2-7B	StreamingLLM	15.09	18.87	29.73	29.60	23.70	8.72	18.69	19.32	20.98	62.50	82.69	34.00	3.00	6.00	55.04	48.80
	H2O	15.63	19.41	30.27	30.14	24.24	9.26	19.23	19.86	21.52	63.00	83.23	34.54	3.50	6.50	55.58	49.34
	HeadKV	15.66	20.29	30.95	30.52	24.50	**9.87**	19.78	20.33	21.93	**64.00**	83.19	**35.18**	**5.00**	5.50	56.01	50.13
	SnapKV	16.10	20.16	30.59	30.44	24.89	9.78	20.15	20.37	21.78	63.50	83.90	34.78	4.50	6.00	56.97	49.98
	PyramidKV	16.32	18.99	30.47	30.65	24.53	9.32	18.95	20.07	22.05	63.00	83.80	34.85	**5.00**	**10.00**	56.97	49.12
	CAKE	15.91	19.69	30.55	30.42	24.52	9.54	19.51	20.14	21.80	63.00	83.51	34.82	4.50	7.00	55.86	49.62
	SCORE(ours)	**17.23**	**21.36**	**35.55**	**30.73**	**25.45**	9.78	**20.55**	**20.51**	**23.85**	**64.00**	**84.30**	35.08	**5.00**	9.50	**57.53**	**51.52**
Llama2-13B	StreamingLLM	12.45	13.66	25.71	11.03	9.27	3.81	20.97	19.47	22.54	68.00	85.42	44.77	4.50	12.00	42.88	41.58
	H2O	12.99	14.20	26.25	11.57	9.81	4.35	21.51	20.01	23.08	68.00	85.96	45.31	4.00	14.75	43.42	42.12
	HeadKV	13.77	15.17	27.50	11.94	10.48	4.01	21.78	20.54	23.57	**69.00**	86.83	46.38	**5.50**	14.25	43.95	42.69
	SnapKV	12.56	12.82	26.13	12.49	10.38	4.85	22.61	20.44	23.56	**69.00**	86.15	45.82	3.05	14.75	**44.33**	43.36
	PyramidKV	**13.85**	15.82	26.33	11.49	9.76	**5.38**	21.35	20.24	23.30	**69.00**	86.09	44.94	5.57	**16.75**	43.18	41.51
	CAKE	13.27	14.48	26.53	11.85	10.09	4.63	21.79	20.29	23.36	68.50	86.24	45.59	4.50	15.25	43.70	42.40
	SCORE(ours)	13.79	**16.61**	**28.99**	**12.75**	**12.36**	4.84	**22.41**	**20.49**	**24.68**	**69.00**	**87.61**	**47.17**	**5.50**	16.50	44.27	**43.39**
Llama3-8B -Instruct	StreamingLLM	22.87	31.41	40.45	43.13	31.64	19.79	20.69	21.66	21.88	70.50	89.70	39.24	4.50	68.48	56.46	52.83
	H2O	23.41	31.95	40.99	43.67	32.18	20.33	21.23	22.20	22.42	71.00	90.24	39.78	5.50	69.02	57.00	53.37
	HeadKV	24.08	34.01	42.64	44.52	33.54	21.31	21.79	22.33	23.09	72.00	90.57	40.62	5.75	69.50	57.68	**55.29**
	SnapKV	23.39	33.50	40.73	43.80	32.84	20.10	21.69	22.57	22.79	71.50	**90.86**	39.85	5.51	**69.50**	**58.14**	53.82
	PyramidKV	23.95	29.53	40.81	43.90	31.37	20.77	21.42	22.89	22.59	71.50	90.48	40.08	**5.91**	69.25	56.37	52.20
	CAKE	23.71	32.25	41.29	43.97	32.48	20.63	21.53	22.50	22.72	71.50	90.54	40.08	5.50	69.32	57.30	53.67
	SCORE(ours)	**25.11**	**35.75**	**42.95**	**45.67**	**35.68**	**21.41**	**22.85**	**22.73**	**24.26**	**72.50**	90.68	**41.99**	5.79	**69.50**	57.35	53.81
Mistral-7B -Instruction-v0.3	StreamingLLM	26.00	30.63	49.01	43.47	31.20	22.43	20.29	18.32	19.01	67.00	84.42	40.70	4.50	79.39	45.22	45.79
	H2O	26.54	31.17	49.55	44.01	31.74	22.97	20.83	18.86	19.55	68.00	84.96	41.24	5.00	79.93	45.76	46.33
	HeadKV	**27.66**	32.30	49.98	44.79	32.35	23.32	20.99	**20.16**	20.03	69.50	85.30	42.35	**6.00**	79.50	46.96	**48.83**
	SnapKV	26.55	31.24	49.58	43.57	31.80	23.14	21.44	18.93	20.03	68.00	85.01	41.36	5.00	81.00	46.45	46.85
	PyramidKV	26.61	31.18	**50.28**	**44.87**	32.28	23.65	21.27	18.69	19.80	67.50	85.78	41.22	5.50	80.50	45.07	44.50
	CAKE	26.34	31.99	49.39	44.02	31.80	23.28	21.28	18.86	19.90	68.00	85.17	41.54	5.00	80.77	46.21	46.26
	SCORE(ours)	27.16	**34.76**	49.82	44.86	**32.57**	**24.27**	**22.34**	20.15	**21.11**	**70.50**	**85.93**	**43.25**	5.50	**82.00**	**48.35**	48.83

Table 15: Performance comparison over 16 datasets from LongBench. Results are measured with a cache size of 256. The best score is marked in **bold**, and the second best is marked with underline.

		Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code	
	Method	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P
Llama2-7B	StreamingLLM	15.94	19.52	33.89	30.44	24.12	9.20	20.32	19.56	22.40	63.00	82.56	34.29	4.00	6.00	56.85	50.41
	H2O	16.48	20.06	34.43	30.98	24.66	9.74	20.86	20.10	22.94	63.00	83.10	34.83	4.50	6.50	57.39	50.95
	HeadKV	17.02	21.38	35.27	**31.48**	25.08	9.50	21.59	**20.76**	22.59	**64.00**	83.02	34.77	**6.00**	5.50	57.98	51.65
	SnapKV	17.02	19.96	34.18	31.37	24.95	10.51	21.74	20.22	23.79	**64.00**	83.57	35.12	5.00	7.00	58.54	50.98
	PyramidKV	16.60	20.03	35.03	31.30	**25.16**	10.40	20.46	20.52	23.63	**64.00**	**83.90**	**35.80**	5.00	8.00	56.85	51.41
	CAKE	16.76	20.34	34.71	31.26	24.94	10.02	21.14	20.38	23.22	63.88	83.38	35.11	4.50	7.05	57.67	51.23
	SCORE(ours)	**18.31**	**21.58**	**35.97**	31.00	25.01	**10.87**	**22.13**	**20.76**	**24.67**	**64.00**	83.75	35.10	5.00	**9.50**	**58.76**	**52.21**
Llama2-13B	StreamingLLM	12.89	14.57	26.60	10.71	10.68	4.05	22.06	19.62	23.88	68.50	84.81	45.60	3.50	15.75	44.09	43.89
	H2O	13.43	15.11	27.14	11.25	11.22	4.59	22.60	20.16	24.42	69.00	85.35	46.14	5.00	16.25	44.63	44.43
	HeadKV	**14.00**	16.05	27.53	11.15	12.32	4.10	22.89	20.52	24.63	**69.50**	85.36	47.19	5.00	15.75	45.10	44.10
	SnapKV	13.53	15.49	26.77	11.74	10.86	5.23	23.58	20.47	25.14	**69.50**	85.68	46.44	5.00	17.25	45.10	46.04
	PyramidKV	13.95	14.99	**28.32**	12.05	11.69	**5.63**	22.52	20.68	24.70	**69.50**	86.21	45.98	5.00	**17.75**	45.10	44.36
	CAKE	13.71	15.39	27.42	11.53	11.50	4.87	22.88	20.44	24.70	69.38	85.63	46.42	**5.50**	16.80	44.91	44.71
	SCORE(ours)	13.65	**16.50**	28.03	**12.51**	**12.45**	4.98	**23.95**	**20.63**	**25.68**	68.50	**87.76**	**47.74**	5.00	**17.75**	**45.71**	**46.99**
Llama3-8B -Instruct	StreamingLLM	24.54	33.31	41.46	43.95	32.30	18.66	22.20	21.90	23.44	71.00	89.60	40.07	5.00	69.25	56.46	53.75
	H2O	25.08	33.85	42.00	44.49	32.84	19.20	22.74	22.44	23.98	71.50	90.14	40.61	5.50	**69.50**	57.00	54.29
	HeadKV	**25.77**	34.67	43.11	45.02	34.04	20.28	22.95	23.02	24.33	73.00	90.56	**41.63**	**6.00**	**69.50**	**57.96**	**55.59**
	SnapKV	25.61	33.63	43.28	44.78	33.78	20.58	23.07	22.62	24.31	71.50	90.44	40.66	**6.00**	68.50	57.01	55.30
	PyramidKV	25.06	34.44	40.82	44.87	31.91	17.93	23.41	22.89	24.49	72.00	90.61	40.74	**6.00**	**69.50**	57.24	53.19
	CAKE	25.38	34.15	42.30	44.79	33.14	19.50	23.04	22.74	24.28	72.07	90.44	40.91	5.50	**69.50**	57.30	54.59
	SCORE(ours)	25.71	**38.27**	**44.30**	**45.82**	**36.54**	**21.11**	**24.54**	**23.11**	**25.49**	**74.00**	**90.65**	**41.63**	5.00	68.50	57.54	54.12
Mistral-7B -Instruction-v0.3	StreamingLLM	26.72	33.71	49.98	44.55	31.82	22.40	22.10	19.27	20.75	68.50	84.74	41.75	4.00	80.00	47.47	48.57
	H2O	27.26	34.25	50.52	45.09	32.36	22.94	22.64	19.81	21.29	69.00	85.28	42.29	4.60	80.50	48.01	49.11
	HeadKV	27.97	35.21	**51.28**	**45.64**	32.90	24.37	22.94	20.94	21.65	**70.50**	85.41	42.80	5.00	82.00	48.97	50.49
	SnapKV	**28.11**	34.54	50.70	45.38	32.92	22.70	23.36	19.89	21.70	70.00	85.50	43.07	5.00	80.50	48.58	49.62
	PyramidKV	26.89	34.21	50.78	45.46	32.46	22.96	22.81	19.80	21.73	68.50	86.13	42.20	5.00	81.00	47.67	48.43
	CAKE	26.87	35.09	50.21	45.06	32.38	23.09	23.05	19.82	21.56	68.93	85.55	42.39	4.50	81.10	48.06	49.13
	SCORE(ours)	27.99	**37.72**	50.85	45.56	**33.02**	**24.84**	**24.21**	**20.99**	**22.46**	69.50	**86.25**	**43.11**	**5.50**	**83.50**	**49.16**	**50.63**

Table 16: Performance comparison over 16 datasets from LongBench. Results are measured with a cache size of 512. The best score is marked in **bold**, and the second best is marked with underline.