R-KV: Redundancy-aware KV Cache Compression for Reasoning Models

Zefan Cai[1✉], **Wen Xiao**[2✉], **Hanshi Sun**[3], **Cheng Luo**[4], **Yikai Zhang**[1], **Ke Wan**[5], **Yucheng Li**[6], **Yeyang Zhou**[5], **Li-Wen Chang**, **Jiuxiang Gu**[7], **Zhen Dong**[8], **Anima Anandkumar**[4], **Abedelkadir Asi**[2], **Junjie Hu**[1✉]

[1]University of Wisconsin - Madison [2]Microsoft [3]Carnegie Mellon University
[4]California Institute of Technology [5]University of California - San Diego [6]University of Surrey
[7]Adobe [8]University of California - Berkeley
https://zefan-cai.github.io/R-KV.page/
https://github.com/Zefan-Cai/R-KV

Abstract

Reasoning models have demonstrated impressive performance in self-reflection and chain-of-thought reasoning. However, they often produce excessively long outputs, leading to prohibitively large key-value (KV) caches during inference. While chain-of-thought inference significantly improves performance on complex reasoning tasks, it can also lead to reasoning failures when deployed with existing KV cache compression approaches. To address this, we propose **R**edundancy-aware **KV** Cache Compression for **R**easoning models (**R-KV**), a novel method specifically targeting redundant tokens in reasoning models. Our method preserves nearly 100% of the full KV cache performance using only 10% of the KV cache, substantially outperforming existing KV cache baselines, which reaches only 60% of the performance. Remarkably, R-KV even achieves 105% of full KV cache performance with 16% of the KV cache. This KV-cache reduction also leads to a 90% memory saving and a 6.6× throughput over standard chain-of-thought reasoning inference. Experimental results show that R-KV consistently outperforms existing KV cache compression baselines across two mathematical reasoning datasets.

1 Introduction

Recent advancements in large language models (LLMs) have demonstrated remarkable capabilities in complex reasoning and self-reflection. However, reasoning models (e.g., DeepSeek-R1 [1]) exhibit a critical deployment challenge: their tendency to produce excessively lengthy and redundant reasoning traces results in unsustainable memory demands [2], primarily due to the rapid growth of the key-value (KV) cache during autoregressive generation. For instance, a DeepSeek-R1-Distill-Llama-8B model may generate 32K tokens to solve a complex math problem, consuming 15.5GB of memory to load the model weight and 4.1GB of memory to store the KV cache. This paradigm of long chain-of-thought (CoT) reasoning generation necessitates the development of KV cache compression.

Outputs from current reasoning models, especially during complex chain-of-thought generation, are fundamentally marked by pervasive redundancy. This inherent characteristic means they are often filled with superfluous content, including unnecessary reflections, iterative re-evaluations, and verbose self-dialogue, all of which add little new semantic value while significantly inflating the

✉ Corresponding to Zefan Cai zefncai@gmail.com, Wen Xiao wxiao@microsoft.com and Junjie Hu junjie.hu@wisc.edu

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

 https://doi.org/10.52202/085713-2038

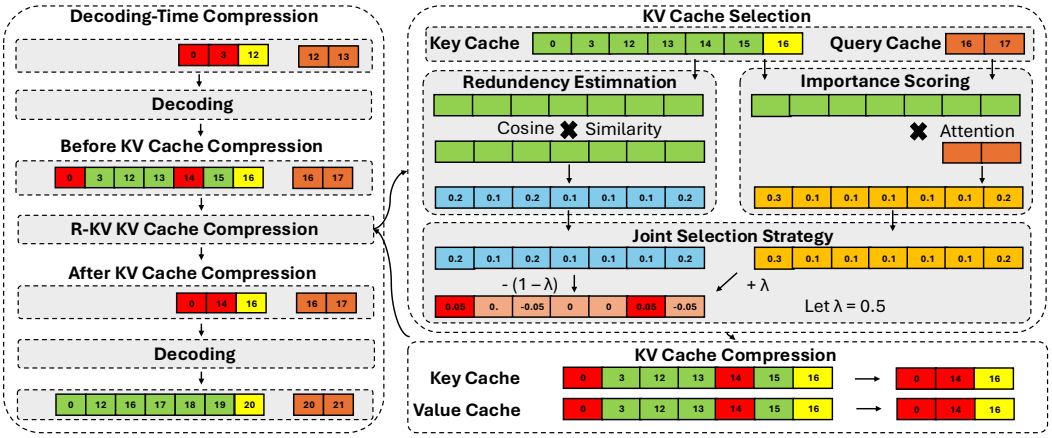


Figure 1: R-KV: (1) Decoding-Time Compression (§3.1); (2) KV Cache Selection with Importance and Redundancy Estimation (§3.2, §3.3) ; (3) KV Cache Compression by joint selection (§3.4).

length of the generation beyond what is needed for concise, effective reasoning. Our analysis (§2.1) shows that over half of the tokens in R1's reasoning chains contribute minimally to task performance, indicating that repetitive self-verification steps or intermediate calculations could be substantially condensed by KV cache compression methods without compromising reasoning accuracy.

However, existing KV cache compression works [3, 4, 5, 6, 7] primarily handle long input prompts but do not explore extensively for long generation outputs. Furthermore, based on our observation (§2.2), standard KV-cache compression methods that rely on simple attention-based importance filtering often fail because the repetitive sections generate high attention signals for themselves. Naively pruning tokens with "low attention weight" may remove crucial but scattered bits of reasoning, or over-retain duplicative self-reflections that appear to have high attention. This observation motivates our exploration of redundancy-aware compression strategies, which selectively retain "important and non-repetitive context" during decoding to preserve the model's critical reasoning ability.

In this work, we propose **R**edundancy-aware **KV** cache compression for reasoning models (i.e., **R-KV**). Our approach consists of three key components: (1) an attention-based importance scoring mechanism that selects critical tokens for retention, (2) a dynamic redundancy scoring mechanism that identifies repetitive tokens through real-time analysis of key vectors, and (3) a joint eviction mechanism that balances both redundancy and importance to optimize cache efficiency.

In our experiments on popular math reasoning benchmarks (§4), by selectively retaining **only 10-34%** of the original KV cache, R-KV achieves comparable performance parity with the uncompressed reasoning model, outperforming state-of-the-art compression baselines with only **60%** of the performance. Remarkably, R-KV even achieves **105%** accuracy of the full KV baseline with around **16%** of the KV cache using DeepSeek-R1-Distill-Llama-8B on the AIME-24 dataset.

This advancement addresses a fundamental tension in deploying state-of-the-art LLMs—balancing reasoning capabilities with practical memory constraints. Our contributions extend beyond technical optimization: we provide systematic evidence that redundancy in CoT generation can be strategically compressed without compromising reasoning abilities. As a training-free and model-agnostic method, R-KV can be used in the rollout process in reinforcement learning (RL) and LLM serving.

2 Observation

2.1 Redundancy in Reasoning Models

As noted in [2], reasoning models often generate a detailed chain of thoughts and multiple reflection steps, resulting in significantly longer responses than standard models. Figure 2 shows that both reasoning models (i.e., DeepSeek-R1-Distill-Llama-8B, DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-14B) generate more than 8× longer generation output compared to the ground truth on two popular math reasoning datasets. However, not all of the additional tokens contribute meaningful content, as much of the decoded context is dominated by repetition. Figure 2

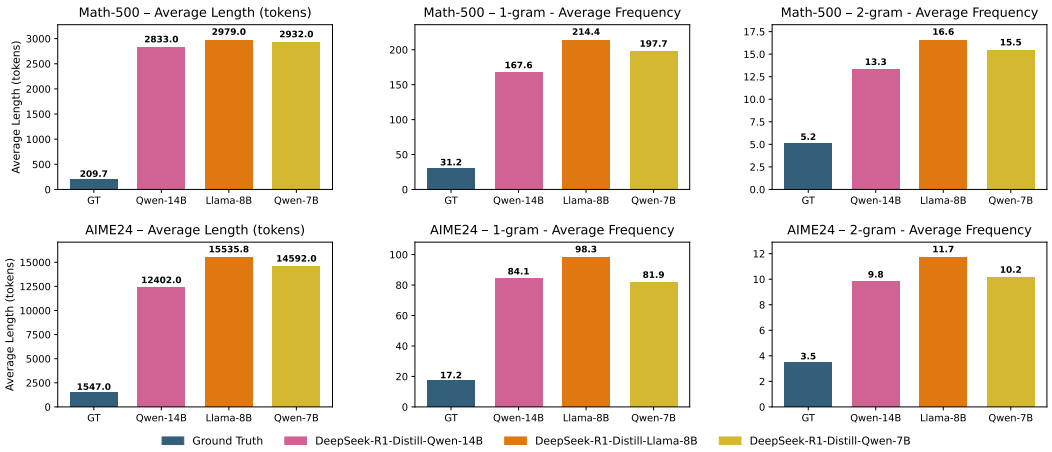


Figure 2: Comparison of generation length and average 1-/2-gram frequency for reasoning models and ground truth of MATH-500 [8] and AIME 2024 [9]. . Reasoning models generate substantially longer responses with 8-14× more tokens, and show higher word repetition with 5-7× higher frequency.

also shows that the average frequency of 1- to 2-grams is consistently higher in the generation output of reasoning models than in ground truth, indicating greater repetitions in the generated outputs of reasoning models.

2.2 Failure of Existing KV Compression Methods to Handle Redundancy

Most existing KV cache compression methods prioritize token selection based primarily on tokens' contextual importance, typically measured through attention scores between key and query tokens [3, 4]. While this approach effectively retains critical context, it fails to account for redundancy—particularly problematic in reasoning models. In such models, we find that repetitive content often receives disproportionately high attention scores, as it closely mirrors previously generated repetitive text. As a result, redundant tokens are excessively retained, unnecessarily inflating the KV cache size without providing additional meaningful new information. In Figure 3, we visualize the cached tokens (inside red boxes) selected by a popular attention-based KV cache method (i.e., SnapKV), showing many repetitions related to self-reflection and conclusion to the final answer.

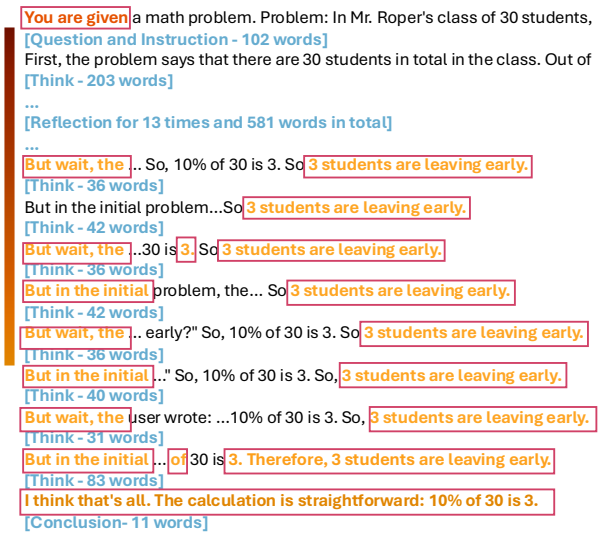


Figure 3: KV selected by SnapKV. SnapKV suffers from redundancy in reasoning models. Black tokens are not selected by SnapKV; brighter colors reflect higher attention scores. Blue tokens are omitted output.

3 Redundancy-aware KV Cache Compression (R-KV)

To address the redundant thinking issue, we propose a *redundancy-aware decoding-time KV cache compression method* (**R-KV**) that explicitly targets the compression of redundant tokens in reasoning models. Our approach balances *importance* and *non-redundancy* in token selection, ensuring that KV cache storage is allocated to both highly informative and diverse content. By incorporating

https://doi.org/10.52202/085713-2038

redundancy estimation into the selection process, our method effectively mitigates unnecessary KV cache growth while preserving the model's reasoning capabilities.

Specifically, R-KV consists of three key components: (1) an *importance scoring mechanism* (§3.2) leveraging attention weights, (2) a *redundancy estimation mechanism* (§3.3) based on semantic similarity of key vectors, and (3) a *joint selection strategy* (§3.4) that optimizes cache efficiency by balancing redundancy and importance.

3.1 Decoding-time Compression

Different from existing KV cache compression methods[3, 5, 4] that focus on the *prefilling stage* to manage long-context inputs, our R-KV focuses on the *decoding stage* for reasoning models—a distinctive setting where the generated output is significantly longer than the prompt.

Specifically, R-KV allocates memory for two components: a cache of budget size B_{budget} to store retained KV tokens, and a buffer of size B_{buffer} for newly generated text tokens. The total memory requirement is thus $B_{\text{total}} = B_{\text{budget}} + B_{\text{buffer}}$. After the model generates each fixed-length text segment in the buffer, R-KV performs KV cache compression. At the end of each text segment, the last α tokens are always retained in the cache as **observation tokens**, following prior work [3]. Next, we concatenate the existing B_{budget} tokens in the cache with the first $B_{\text{buffer}} - \alpha$ tokens in the buffer, resulting in $n = B_{\text{budget}} + B_{\text{buffer}} - \alpha$ candidate KV tokens. Each candidate is assigned a selection score (§3.4), and we select the top $k = B_{\text{budget}} - \alpha$ tokens to fit in the rest of the cache budget, in addition to the α observation tokens. This process compresses the KV cache while preserving critical context, enabling efficient memory utilization during autoregressive decoding.

3.2 Importance Scoring via Attention Weights

Following attention-based methods (e.g., SnapKV [3], PyramidKV [5]), R-KV estimates token importance using attention weights, leveraging the intuition that tokens receiving higher attention contribute more to decoding and are thus more critical for preserving model performance. Specifically, we compute each key token's attention scores received from the last α **observation tokens** during decoding. In addition to the standard multi-head attention mainly adopted by the prior works [3], we also propose the importance score estimation using the grouped-query attention. Below, we detail the estimation on top of these two popular attention mechanisms used by current LLMs.

Multi-Head Attention (MHA). Given the last α observation tokens as query $\boldsymbol{Q}^h \in \mathbb{R}^{\alpha \times d}$ and n key states $\boldsymbol{K}^h \in \mathbb{R}^{n \times d}$ for each attention head h, the attention scores $\boldsymbol{A}^h \in \mathbb{R}^{\alpha \times n}$ are computed as:

$$\boldsymbol{A}^h = \text{softmax}(\boldsymbol{Q}^h \cdot (\boldsymbol{K}^h)^\top / \sqrt{d}). \tag{1}$$

Grouped-Query Attention (GQA). In GQA, each key/value head h is shared among a group of G distinct query heads indexed by $g \in [0, G)$. Correspondingly, we denote the shared key/value states as $\boldsymbol{K}^h, \boldsymbol{V}^h \in \mathbb{R}^{n \times d}$, and the G query states as $\boldsymbol{Q}^{h,0}, \ldots, \boldsymbol{Q}^{h,G-1} \in \mathbb{R}^{\alpha \times d}$ within the head group indexed by h, where n is the number of key/value states, d is the head hidden dimension. The attention score for each of the G query heads within the group is computed as:

$$\boldsymbol{A}^{h,g}_{\text{group}} = \boldsymbol{Q}^{h,g} \cdot (\boldsymbol{K}^h)^\top / \sqrt{d} \in \mathbb{R}^{\alpha \times n}, \quad \text{for } g = 0, \ldots, G-1. \tag{2}$$

These G individual matrices are then aggregated into a single consolidated matrix $\boldsymbol{A}^h_{\text{group}}$ for the head group h using a max-pooling operation across the group dimension. The final attention weight $\boldsymbol{A}^h$ for the head group h is then obtained by renormalizing $\boldsymbol{A}^h_{\text{group}}$ along the key token dimension.

$$\boldsymbol{A}^h_{\text{group}} = \text{maxpool}\left(\left[\boldsymbol{A}^{h,0}_{\text{group}}, \ldots, \boldsymbol{A}^{h,G-1}_{\text{group}}\right]\right) \in \mathbb{R}^{\alpha \times n}, \quad \boldsymbol{A}^h = \text{softmax}\left(\boldsymbol{A}^h_{\text{group}}\right) \in \mathbb{R}^{\alpha \times n} \tag{3}$$

Stabilization and Importance Estimation. We use $\boldsymbol{A}^h$ hereafter to denote the attention weights calculated using either MHA or GQA. Note that the per-token importance scores derived from $\boldsymbol{A}^h$ may contain outliers with excessively high values, resulting in unstable estimation of importance scores. To mitigate this influence, we follow the prior work [3] and apply a max-pooling operation to these per-token importance scores over a sliding window of size $2W$ across recent tokens. Specifically, we denote $A^h_{j,i}$ as the attention score from the j-th query to the i-th key in $\boldsymbol{A}^h$. We obtain the stabilized

attention score $\tilde{\boldsymbol{A}}^h$ by computing its (i, j) entry, and finally obtain the importance score of retaining the i-th token in the KV cache as I_i^h for each attention head h, as shown below:

$$\tilde{A}^h_{j,i} = \max\left(A^h_{j,i-W}, \ldots, A^h_{j,i}, \ldots, A^h_{j,i+W-1}\right), \quad I^h_i = \frac{1}{\alpha}\sum_{j=0}^{\alpha-1} \tilde{A}^h_{j,i} \in \mathbb{R}. \tag{4}$$

3.3 Redundancy Estimation via Semantic Similarity

To identify redundant tokens, we measure the semantic similarity between key states using cosine similarity. Tokens with high similarity to others are considered potentially redundant and can be selectively removed to optimize KV cache memory.

Cosine Similarity between Key Tokens: Given the key tokens $\boldsymbol{K}^h \in \mathbb{R}^{n\times d}$ for a specific head h, We first normalize each key vector $\boldsymbol{K}^h_i, \forall i \in [0, 1)$ into $\overline{\boldsymbol{K}}^h_i$, and then compute the cosine similarity matrix $\boldsymbol{S}^h$ using the normalized key vectors.

$$\overline{\boldsymbol{K}}^h_i = \frac{\boldsymbol{K}^h_i}{\|\boldsymbol{K}^h_i\|_2 + \epsilon} \in \mathbb{R}^d, \quad \boldsymbol{S}^h = \overline{\boldsymbol{K}}^h(\overline{\boldsymbol{K}}^h)^\top \in \mathbb{R}^{n\times n}, \quad S^h_{i,i} \leftarrow 0, \forall i \in [0, n), \tag{5}$$

where $\|\cdot\|_2$ is the L2 norm and ϵ is a small constant (e.g., 10^{-8}) for numerical stability. To prevent tokens from being marked as redundant with themselves, we zero out the diagonal elements $S^h_{i,i}$.

Enforce Retention of Recent Tokens. While redundant, such tokens may still carry meaningful information. Thus, naively removing all redundant tokens can impair model performance. To address this, we retain only the β most recently generated tokens among those exhibiting high similarity, as these later tokens tend to better support the model's decoding than earlier ones. To enforce this, we further zero out the similarity scores in $\boldsymbol{S}^h$ corresponding to these β most recent similar tokens. Formally, for each token $i \in [0, n)$, we identify the set of indices of highly similar tokens: $\mathcal{I}^h_i = \{j \mid S^h_{j,i} > T, j \in [0, n)\}$, where T is a fixed hyperparameter for similarity threshold. For this set, we extract the subject $\mathcal{I}^h_{i,\beta} \subseteq \mathcal{I}^h_i$, containing up to the β largest indices—i.e., the β most recent similar tokens to token i, or fewer if not enough such tokens exist. We then suppress their influence by zeroing out their similarity scores with token i in $\boldsymbol{S}^h$, i.e., $S^h_{j,i} \leftarrow 0,\ \forall j \in \mathcal{I}^h_{i,\beta}$. This modification effectively nullifies the direct similarity links from token i to its β most recent highly similar tokens.

Redundancy Score Estimation: Finally, we compute normalized redundancy scores for all key tokens in Eq. (6). First, for each key token $i \in [0, n)$ in each head h, we compute its average similarity score $\bar{S}^h_i$. Intuitively, $\bar{S}^h_i$ measures how similar token i is, on average, to all other key tokens in the sequence. A high $\bar{S}^h_i$ indicates that the semantic content of token i is largely shared with other tokens, suggesting potential redundancy. Next, to obtain per-token redundancy scores R^h_i within a fixed numerical range for each head h, we normalize $\bar{S}^h_i$ using a softmax operation. The resulting score R^h_i reflects the redundancy of token i for head h, with higher values indicating greater redundancy.

$$\bar{S}^h_i = \frac{1}{n}\sum_{j=0}^{n-1} S^h_{j,i}, \quad R^h_i = \left(\text{softmax}\left([\bar{S}^h_0, \ldots, \bar{S}^h_{n-1}]\right)\right)_i \tag{6}$$

3.4 Joint Selection Strategy for KV Cache Retention

To efficiently manage KV cache storage while retaining essential context, we employ a joint selection strategy that integrates both importance and redundancy scores. Given a predefined token budget B_{budget} per attention head, our goal is to retain tokens that maximize information diversity while minimizing redundancy. The final selection score Z^h_i for each token i in head h is computed as:

$$Z^h_i = \lambda I^h_i - (1 - \lambda)\, R^h_i, \tag{7}$$

where the importance score I^h_i and the redundancy score R^h_i are computed in Eq. (4) and Eq. (6) respectively. A higher I^h_i indicates that a token is more important and should ideally be retained, while a higher R^h_i suggests higher token redundancy. The hyperparameter λ controls the trade-off

 https://doi.org/10.52202/085713-2038

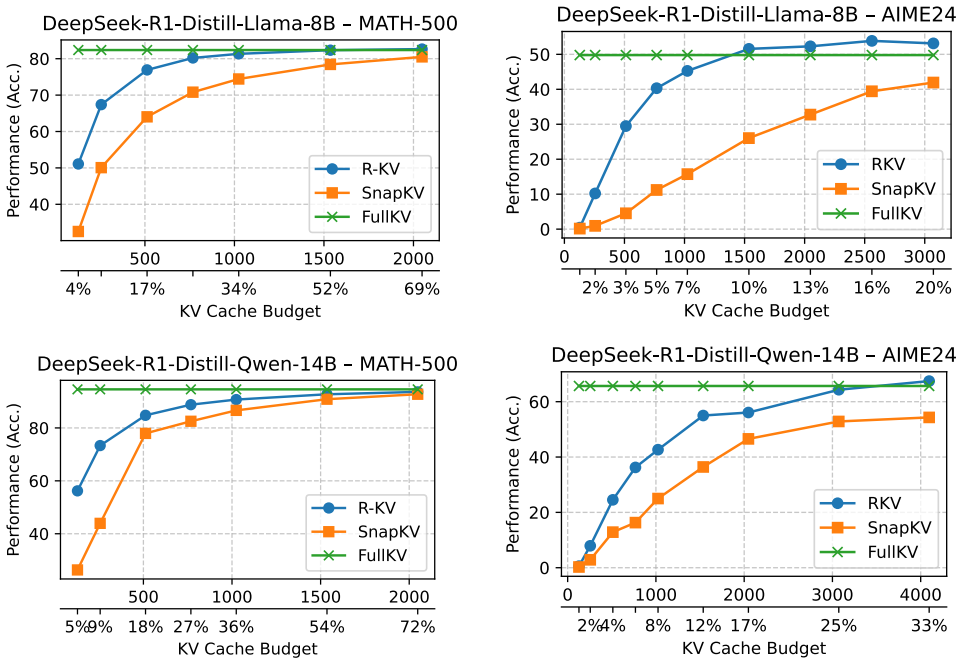


Figure 4: Results of R-KV compared with SnapKV and FullKV on the MATH-500 and AIME24 datasets for R1-Llama-8B (**top**) and R1-Qwen-14B (**bottom**). Results are reported as pass@1 based on 64 generated responses per question.

between prioritizing important tokens and reducing redundant tokens. We discuss the rationale for choosing λ through a sensitivity analysis in §5.1. This strategy ensures that the KV cache prioritizes storing tokens that are both important and semantically diverse, thereby improving memory efficiency without compromising model performance.

4 Experiment

4.1 Experimental Setup

Models and Datasets In our experiments, we use variants of the DeepSeek-R1 distilled model: DeepSeek-R1-Distill-Llama-8B, and DeepSeek-R1-Distill-Qwen-14B [1], which we refer to as R1-Llama-8B and R1-Qwen-14B, respectively, for brevity throughout the paper.

We evaluate the models' mathematical reasoning capabilities using three benchmarks: MATH-500 [8] and AIME 2024 [9].

Hyperparameters We set $B_{\text{buffer}} = 128$, $\alpha = 8$ and $\lambda = 0.1$, with an analysis of λ in §5.1.

Baselines We compare our method against SnapKV [3], originally designed for long prefilling. To adapt it for decoding, we apply the same compression interval as our method, i.e., compressing the KV cache every 128 decoding steps using identical B_{budget} and B_{buffer}. Our approach focuses on improving KV cache eviction through a hybrid strategy, and we therefore restrict comparison to state-of-the-art attention-based eviction methods. Budget allocation techniques (e.g., head-level [6] and layer-level [5]) are orthogonal to our work and not included. We also report results for FullKV, which retains the full KV cache and serves as the gold standard for decoding quality.

Evaluation Setup We set the maximum generation length to 16,384 tokens for MATH-500 and 32,768 tokens for AIME 2024 and AIME 2025, because further increasing the generation length has shown no improvement on model performance on these datasets from our attempts. We find that using greedy decoding to evaluate long-output reasoning models results in significant variability across different setups. Following existing works [1], we utilize pass@k evaluation [10] and report

pass@1 using a non-zero temperature. We use the recommended sampling temperature and top-p value for each model, i.e., sampling temperature of 0.6 and a top-p value of 0.95 for DeepSeek-R1 Distilled models. We generate 64 responses for each question. Pass@1 is then calculated as Pass@1 $= \frac{1}{k}\sum_{i=1}^{k} p_i$, where p_i denotes the correctness of the i-th response. This method provides more reliable performance estimates.

4.2 Results

The accuracy performance of R-KV compared with all baselines is shown in Figure 4, with detailed accuracy numbers in Appendix B.2. The KV cache budget ratio is calculated based on the KV cache budget and the average generation length of tokens, i.e., R1-Llama-8B: $2,979.1$ on MATH-500 and $15,535.8$ on AIME24; R1-Qwen-14B: $2,833.04$ on MATH-500 and $12,402$ on AIME24. Our method significantly outperforms the baseline SnapKV, achieving up to 40% Acc. improvement. We provide two KV cache budget and performance analysis. Fixed budget analysis is more practical because when the model outputs longer (i.e., from $2,979.1$ on MATH-500 to $15,535.8$ on AIME24), the KV cache budget needed for lossless compression increases less (i.e., 512). In the KV cache budget ratio perspective, the changes of lossless compression ratio is dominated by generation length.

Ratio Budget For R1-Llama-8B, R-KV achieves lossless compression with 34% KV cache budget on the MATH-500 dataset and with 10% KV cache budget on the AIME-2024 dataset. Given 16% KV cache budget, our method even surpasses the FullKV baseline, reaching 105% of its accuracy. Similarly, for R1-Qwen-14B, R-KV achieves lossless compression with 54% KV cache budget on the MATH-500 dataset and with 25% KV cache budget on the AIME-2024 dataset. Given 33% KV cache budget, our method achieves 105% of FullKV accuracy.

Fixed Budget For R1-Llama-8B, R-KV achieves lossless compression with 1024 KV cache budget on the MATH-500 dataset and with 1536 KV cache budget on the AIME-2024 dataset. For R1-Llama-8B, R-KV achieves lossless compression with 1536 KV cache budget on the MATH-500 dataset and with 3072 KV cache budget on AIME-2024.

5 Discussion

5.1 How to Choose λ?

Figure 5 shows the distributions of the Importance Score (I^h) and Redundancy Estimation (R^h) for head $h = 0$ at the top layer ($N_{\text{layer}} = 31$). The figure reveals that I^h is sparse and dominated by a few outlier values, while the similarity distributions (which inform R^h) are relatively dense. When $\lambda = 0$, the token retention strategy is overned entirely by Redundancy Estimation (R^h). As shown in Figure 5, the initial four tokens are not guaranteed to be preserved. As highlighted by prior work [7], evicting these initial tokens can severely impair the generative capabilities of LLMs. Therefore, it is crucial to select a λ value that starts from at least 0.01. On the other hand, as λ increases beyond 0.1, the selection metric becomes increasingly dominated by attention scores. These observations suggest that an optimal λ lies within the range of $0.01 \leq \lambda \leq 0.1$, effectively balancing the contributions of Importance Score and Redundancy Estimation.

Figure 6 presents the accuracy (Acc.) performance of R-KV on the DeepSeek-Distill-R1-Llama-8B model using the MATH-500 dataset. The results further guide the choice of λ for optimal performance. The figure demonstrates that $\lambda = 0.1$ yields the highest accuracy. In contrast, strategies relying solely on redundancy ($\lambda = 0$) or solely on attention ($\lambda = 1$) exhibit the poorest performance, underscoring the complementary nature of these two metrics and the importance of a balanced approach. Thus, based on this finding, we select $\alpha = 0.1$ for all evaluations detailed in Figure 4.

5.2 Failure of Attention-Based Methods to Capture Redundancy

To thoroughly investigate the advantages of R-KV's hybrid selection metrics (combining attention and redundancy) over pure attention-based importance metrics, we compared the tokens selected by R-KV against those chosen by a pure attention-based method (SnapKV). We present a case where R-KV correctly completes the task while the comparison method fails. As illustrated in Figure 7,

 https://doi.org/10.52202/085713-2038

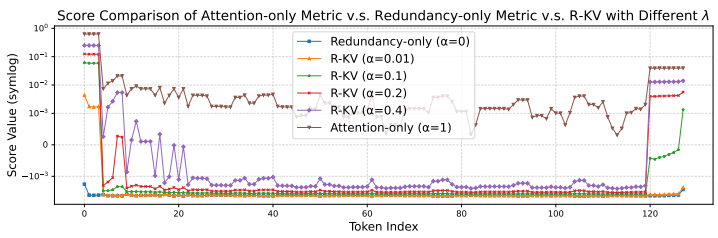


Figure 5: KV sekection score comparison of attention-only metric v.s. redundency-only metric v.s. R-KV with different λ. When $\lambda \geq 0.1$, the selection score starts to be dominated by attention score.

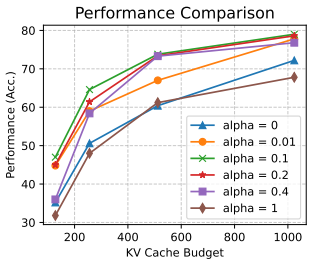


Figure 6: Performance Comparison of the same methods as Figure 5.



(a) SnapKV (b) R-KV

Figure 7: Comparison of selected key-value (KV) tokens for an example between SnapKV (left) and R-KV (right). Grey tokens are unselected, while the gradient from light to dark red indicates the number of attention heads selecting each token (darker = more heads). R-KV selects a more diverse and broadly distributed set of tokens, capturing richer contextual information.

grey tokens represent unselected tokens, while the gradient from light orange to red indicates the number of heads selecting each token, with darker red signifying selection by more heads.

When considering the tokens selected by all heads, we observe that R-KV selects a more diverse set of tokens that cover a broader range and contain more effective information. These selections are more evenly distributed throughout the decoded output, capturing a more comprehensive context representation. In contrast, SnapKV's selected tokens exhibit more limited coverage. It tends to favor tokens positioned close to the query token, which are often selected multiple times by various heads, indicating a concentration of attention in localized areas. Furthermore, SnapKV also selects tokens that are not in close proximity to the query but still constitute largely redundant and unimportant segments (i.e., "3 students are leaving early." and "But in the initial").

5.3 Efficiency Analysis

Memory Saving R-KV achieves improved memory efficiency by allocating fixed-size buffers for both the retained KV cache and newly generated tokens. Unlike FullKV, which scales memory linearly with sequence length, R-KV 's memory footprint remains constant, enabling substantial savings during long-form generation. Detailed memory accounting is provided in Appendix C.1.

Gen. Length	Method	Budget	Mem. Saving (%)	Batch	Throughput (tok/s)	Tokens Gen.	Dec. Time (s)
8K	FullKV	–	–	1	75.44	8 094	107.30
		–	–	62 (max)	849.13	501 828	590.99
	R-KV	Fixed – 1024	87.50	1	80.46	8 094	100.60
		Fixed – 1024	87.50	402 (max)	3 251.52	3 253 788	1 000.70
		Fixed – 1536	81.25	287 (max)	2 525.75	6 546 972	919.72
		Ratio – 10% – 819	90.00	479 (max)	3 809.15	3 877 026	1 017.82
		Ratio – 34% – 2 785	66.00	167 (max)	1 608.01	1 351 698	840.61
		Ratio – 54% – 4 423	46.00	105 (max)	1 257.83	849 870	675.66
16K	FullKV	–	–	1	69.41	16 286	234.65
		–	–	30 (max)	347.03	488 580	1 407.89
	R-KV	Fixed – 1024	93.75	1	80.95	16 286	201.18
		Fixed – 1024	93.75	402 (max)	3 188.82	6 546 972	2 053.10
		Fixed – 1536	90.63	287 (max)	2 447.61	4 674 082	1 909.65
		Ratio – 10% – 1 638	90.00	271 (max)	2 300.28	4 413 506	1 918.68
		Ratio – 34% – 5 570	66.00	82 (max)	797.43	1 335 452	1 674.70
		Ratio – 54% – 8 847	46.00	46 (max)	584.77	749 156	1 281.12

Table 1: Memory saving, throughput, and decoding-time comparison for Llama3-8B under various generation length and KV cache compression budget settings.

Computation Overhead While R-KV introduces additional computation for importance and redundancy scoring, the total overhead is modest and often outweighed by the reduced attention cost over a compressed KV cache. This trade-off becomes increasingly favorable as sequence length grows. Complexity comparisons can be found in Appendix C.1

Real-time analysis We present the real-time analysis of memory saving and end-to-end throughput improvement in Table 1. When the batch size is 1, R-KV exhibits a slight throughput advantage over FullKV. This suggests that the acceleration achieved by R-KV through reduced attention computation outweighs computational overhead of R-KV. However, this direct speedup constitutes a minor portion of the overall benefit. The primary throughput improvement from R-KV stems from enabling significantly larger inference batch sizes due to KV cache compression.

We evaluate end-to-end throughput under both ratio-based and fixed KV cache budgets. R-KV consistently enables much larger batch sizes and higher throughput than FullKV, with benefits becoming more pronounced at longer sequence lengths. For example, at a sequence length of 16K, R-KV achieves up to 9× larger batch sizes and over 6.6× higher throughput under a 10% compression ratio, and 13.4× larger batch sizes with 9.2× throughput under a fixed budget of 1024. Detailed analysis are provided in Appendix C.2.

6 Related Work

KV Cache Compression The optimization of KV cache memory efficiency in LLMs has garnered increasing attention as model sizes and context windows expand. Existing approaches primarily fall into three categories: dynamic token eviction[3, 11, 12], quantization[13, 14, 15], merging[16, 17, 18], and low-rank decomposition[19, 20, 21]. Previous eviction methods like SnapKV[3], PyramidKV[5], Ada-KV[22], HeadKV[6] dynamically prune tokens based on attention scores, but mainly focus on evicting tokens for prefilling stage. StreamingLLM[7] and H2O[4] are proposed for decoding. However, these general-purpose techniques often struggle with reasoning-intensive tasks, where aggressive eviction risks disrupting critical intermediate steps in CoT, and suffers from reasoning models' inherent redundancy.

Efficient Reasoning Recent works in efficient reasoning focus on training the model to generate less CoT without sacrificing performance. [23, 24, 25] use RL optimization with length penalty rewards to encourage models to produce more concise chains-of-thought (CoT). [26, 27] employs variable-length CoT datasets to supervised fine-tune (SFT) the LLM to reduce token usage while preserving reasoning correctness. Both RL and SFT methods require additional training. [27, 28, 29] use test-time prompting to reduce generation length, but these methods may hurt the performance. As a KV cache compression work for reasoning models, R-KV is able to achieve lossless compression without extensive training and prompting.

https://doi.org/10.52202/085713-2038

7 Conclusion

We introduced R-KV, a novel decoding-time KV cache compression method tailored to the challenges of complex reasoning in large language models (LLMs). Reasoning models often generate long, redundant outputs that impose substantial memory and computational burdens during inference. R-KV addresses this by jointly scoring token importance and redundancy, enabling the retention of essential reasoning content while discarding repetitive or uninformative tokens. This dynamic and attention-guided strategy allows R-KV to preserve nearly full model performance using only 10–34% of the original KV cache—substantially outperforming prior compression methods.

Extensive throughput and efficiency analysis demonstrate that R-KV enables up to 13× larger batch sizes and over 9× speedup in long-sequence generation scenarios compared to FullKV, with particularly strong gains under constrained memory budgets. With its training-free and model-agnostic design, R-KV provides a scalable and deployment-ready solution for reasoning LLMs, especially in streamlining the rollout phase of reinforcement learning workflows.

8 Acknowledgement

Research reported in this publication was partially supported by the National Science Foundation under Award Number IIS-2449768. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Science Foundation.

References

[1] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[2] Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning, 2025.

[3] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, T sianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation, 2024.

[4] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023.

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

[6] 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, 2024.

[7] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024.

[8] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[9] MAA. American invitational mathematics examination - aime. In *American Invitational Mathematics Examination - AIME 2024*, February 2024.

 https://doi.org/10.52202/085713-2038

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[11] 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. *arXiv preprint arXiv:2310.01801*, 2023.

[12] 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.

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

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

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

[16] Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient llms inference. In *Forty-first International Conference on Machine Learning*, 2024.

[17] Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. *arXiv preprint arXiv:2312.03414*, 2023.

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

[19] Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference, 2025.

[20] Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for kv cache compression, 2024.

[21] Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. Lorc: Low-rank compression for llms kv cache with a progressive compression strategy, 2024.

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

[23] Chen Li, Nazhou Liu, and Kai Yang. Adaptive group policy optimization: Towards stable training and token-efficient reasoning, 2025.

[24] Junjie Yang, Ke Lin, and Xing Yu. Think when you need: Self-adaptive chain-of-thought learning, 2025.

[25] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with llms, 2025.

[26] Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, Yue Xing, Jiliang Tang, and Qi He. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models, 2025.

[27] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning, 2025.

[28] Yule Liu, Jingyi Zheng, Zhen Sun, Zifan Peng, Wenhan Dong, Zeyang Sha, Shiwen Cui, Weiqiang Wang, and Xinlei He. Thought manipulation: External thought can be efficient for large reasoning models, 2025.

[29] Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking, 2025.

[30] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao

 https://doi.org/10.52202/085713-2038

Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez,

Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

[31] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.

[32] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore, December 2023. Association for Computational Linguistics.

A Method

A.1 Algorithm

The pseudo-code of the method is shown in Algorithm 1.

A.2 Implementation Details

Max Pooling of Attention Weights Latest open-source LLMs [30, 31] have widely adopted Grouped-Query Attention (GQA) [32], where multiple query heads share a common pair of key-value heads to substantially reduce memory access overhead during inference. In key-value (KV) cache eviction strategies, it's thus often necessary to downscale attention scores from (Q_head, seq_len, seq_len) to (KV_head, seq_len, seq_len). While previous works such as SnapKV [3] have predominantly employed mean pooling to aggregate attention scores across query head groups, we hypothesize that max pooling could better preserve the most critical tokens for each query head. Our empirical results demonstrate that max pooling leads to improved performance, and we adopt it for all main experiments.

Calibration of SnapKV's Observation Window Mask The official implementation of SnapKV applied an upper triangular attention mask to the attention weights matrix to enforce causality. The attention weights matrix is then processed with softmax, slicing, and summation to obtain observation window scores for each prefix token.

They adopted an upper triangle to prevent tokens in the observation window from seeing tokens after them, and then applied softmax and summation. However, their implementation does not account for the fact that tokens within the observation window absorb part of the attention weight originally assigned to prefix tokens, thereby disrupting the normalization property.

In our implementation, we address this issue by first slicing the attention weights before applying softmax. This approach ensures proper normalization and leads to significantly better scores in our tests.

```
if not fix_obs_window:
    mask = torch.full(
```

 https://doi.org/10.52202/085713-2038

Algorithm 1 R-KV: Q_{obs} are query states for α observation tokens, $K_{\text{full}}, V_{\text{full}}$ are the full KV cache states of length L_{full}.

procedure R-KV($(\boldsymbol{K}_{\text{full}}, \boldsymbol{V}_{\text{full}}), L_{\text{full}}, L_{\text{budget}}, Q_{\text{obs}}, \alpha, B_{\text{budget}}, B_{\text{buffer}}, T, \beta, \lambda, \epsilon, H, d_k$)
  if $L_{\text{full}} - L_{\text{budget}} < B_{\text{buffer}}$ **then** ▷ Check if compression is triggered
    return $(\boldsymbol{K}_{\text{full}}, \boldsymbol{V}_{\text{full}})$
  end if
  $(\boldsymbol{K}_{\text{obs}}, \boldsymbol{V}_{\text{obs}}) \leftarrow$ last α tokens of $(\boldsymbol{K}_{\text{full}}, \boldsymbol{V}_{\text{full}})$
  $(\boldsymbol{K}_{\text{cand}}, \boldsymbol{V}_{\text{cand}}) \leftarrow$ first $(L_{\text{full}} - \alpha)$ tokens of $(\boldsymbol{K}_{\text{full}}, \boldsymbol{V}_{\text{full}})$
  $N_c \leftarrow L_{\text{full}} - \alpha$ ▷ Number of candidate tokens
  if $N_c \leq B_{\text{budget}}$ **then**
    return $(\boldsymbol{K}_{\text{full}}, \boldsymbol{V}_{\text{full}})$ ▷ Not enough candidates to prune beyond budget
  end if
  for each head $h = 0 \dots H-1$ **do**
    Compute attention matrix $\boldsymbol{A}^h \in \mathbb{R}^{\alpha \times N_c}$ using Q^h_{obs} and $\boldsymbol{K}^h_{\text{cand}}$ ▷ Handles MHA/GQA as per Eqs. (1)-(3) from text
    for $k = 0 \dots N_c - 1$ **do** ▷ For each candidate token k
      $I'_{k,h} \leftarrow \frac{1}{\alpha} \sum_{q=0}^{\alpha-1} (\boldsymbol{A}^h)_{qk}$ ▷ q: observation token, k: candidate token
    end for
    $\{I_{k,h}\}_{k=0}^{N_c-1} \leftarrow$ 1D-Pooling$(\{I'_{k,h}\}_{k=0}^{N_c-1})$
  end for
  for each head $h = 0 \dots H-1$ **do**
    $\boldsymbol{K}^h_{\text{norm}} \in \mathbb{R}^{N_c \times d_k}$; For $k = 0 \dots N_c - 1$, $\boldsymbol{K}^h_{\text{norm},k} \leftarrow \boldsymbol{K}^h_{cand,k} / (\|\boldsymbol{K}^h_{cand,k}\|_2 + \epsilon)$
    $\boldsymbol{S}^h \leftarrow \boldsymbol{K}^h_{\text{norm}} (\boldsymbol{K}^h_{\text{norm}})^\top$ ▷ Cosine Similarity Matrix Computation, similarity matrix $\boldsymbol{S}^h \in \mathbb{R}^{N_c \times N_c}$
    for $k = 0 \dots N_c - 1$ **do** ▷ Prevent Self-Redundancy
      $(\boldsymbol{S}^h)_{kk} \leftarrow 0$
    end for
    $B^h_{uv} \leftarrow ((\boldsymbol{S}^h)_{uv} > T ? 1 : 0)$ for $u, v \in \{0, \dots, N_c - 1\}$ ▷ Identify Highly Similar Pairs
    for $u = 0 \dots N_c - 1$ **do** ▷ Enforce Retention of Recent Tokens
      $T^h_u \leftarrow \{v \mid B^h_{uv} = 1, v \in \{0, \dots, N_c - 1\}\}$
      $T^h_{u,\beta} \leftarrow$ subset of T^h_u with up to β largest indices v.
      for $v' \in T^h_{u,\beta}$ **do**
        $(\boldsymbol{S}^h)_{u,v'} \leftarrow 0$
      end for ▷ $\boldsymbol{S}^h$ is now modified
    end for
    Let $\bar{\mathbf{S}}^h \in \mathbb{R}^{N_c}$ where $(\bar{\mathbf{S}}^h)_u \leftarrow \frac{1}{N_c} \sum_{v=0}^{N_c-1} (\boldsymbol{S}^h)_{uv}$
    for $u = 0 \dots N_c - 1$ **do**
      $R_{u,h} \leftarrow (\text{softmax}(\bar{\mathbf{S}}^h))_u$
    end for
  end for
  for each head $h = 0 \dots H-1$ **do**
    for $k = 0 \dots N_c - 1$ **do**
      $\text{Score}_{k,h} \leftarrow \lambda I_{k,h} - (1-\lambda) R_{k,h}$
    end for
  end for
  Let $\text{AggScore} \in \mathbb{R}^{N_c}$
  for $k = 0 \dots N_c - 1$ **do**
    $\text{AggScore}_k \leftarrow \text{mean}_h(\text{Score}_{k,h})$ ▷ Aggregate scores across heads
  end for
  $Idx_{sel} \leftarrow$ indices of top-B_{budget} tokens from $\{0, \dots, N_c - 1\}$ based on AggScore
  $\boldsymbol{K}_{cand_sel} \leftarrow \boldsymbol{K}_{cand}[Idx_{sel}]$; $\boldsymbol{V}_{cand_sel} \leftarrow \boldsymbol{V}_{cand}[Idx_{sel}]$
  $\boldsymbol{K}_{comp} \leftarrow \text{concatenate}(\boldsymbol{K}_{cand_sel}, \boldsymbol{K}_{obs})$ ▷ Order might vary
  $\boldsymbol{V}_{comp} \leftarrow \text{concatenate}(\boldsymbol{V}_{cand_sel}, \boldsymbol{V}_{obs})$
  $L_{prev_comp} \leftarrow B_{budget} + \alpha$ ▷ Update length for next cycle
  return $(\boldsymbol{K}_{comp}, \boldsymbol{V}_{comp})$
end procedure

Model	Benchmark	Method	128	256	512	768	1 024	1 536	2 048	2 560	3 072	4 096
Llama3-8B	MATH	FullKV	82.38	82.38	82.38	82.38	82.38	82.38	82.38	–	–	–
		R-KV	51.08	67.39	76.92	80.21	81.34	82.34	82.65	–	–	–
		SnapKV	32.53	50.07	64.03	70.81	74.43	78.43	80.50	–	–	–
	AIME24	FullKV	49.79	49.79	49.79	49.79	49.79	49.79	49.79	49.79	49.79	–
		R-KV	0.42	10.21	29.48	40.31	45.26	51.56	52.29	53.85	53.13	–
		SnapKV	0.16	0.94	4.53	11.20	15.73	26.04	32.76	39.43	41.93	–
Qwen-14B	MATH	FullKV	94.58	94.58	94.58	94.58	94.58	94.58	94.58	–	–	–
		R-KV	56.21	73.33	84.77	88.79	90.72	92.72	93.62	–	–	–
		SnapKV	26.32	43.93	77.93	82.52	86.63	90.86	92.73	–	–	–
	AIME24	FullKV	65.68	65.68	65.68	65.68	65.68	65.68	65.68	–	65.68	65.68
		R-KV	0.57	7.92	24.53	36.25	42.66	55.00	56.09	–	64.32	67.45
		SnapKV	0.26	2.86	12.86	16.30	25.00	36.41	46.56	–	52.86	54.32

Table 2: Accuracy (%) of **Llama3-8B** and **Qwen-14B** on the MATH and AIME24 benchmarks under different memory-optimization methods across context lengths. "–" denotes configurations that were not evaluated.

```
        (self.window_size, self.window_size),
        torch.finfo(attn_weights.dtype).min,
        device=attn_weights.device,
    )
    mask_cond = torch.arange(mask.size(-1), device=attn_weights.device)
    mask.masked_fill_(mask_cond < (mask_cond + 1).view(mask.size(-1), 1), 0)
    mask = mask.to(attn_weights.device)
    attention_mask = mask[None, None, :, :]
    attn_weights[
        :, :, -self.window_size :, -self.window_size :
    ] += attention_mask

    attn_weights = nn.functional.softmax(
        attn_weights, dim=-1, dtype=torch.float32
    ).to(query_states.dtype)

    attn_weights_sum = attn_weights[
        :, :, -self.window_size :, : -self.window_size
    ].mean(dim=-2)
else:
    attn_weights_sum = (
        nn.functional.softmax(
            attn_weights[:, :, -self.window_size :, : -self.window_size],
            dim=-1,
            dtype=torch.float32,
        )
        .mean(dim=-2)
        .to(query_states.dtype)
    )
```

B Experiment

B.1 Devices

We use NVIDIA A100 80G to finish all the experiments.

B.2 Main Results

See Table 2.

https://doi.org/10.52202/085713-2038

C Efficiency

C.1 Complexity Analysis of Memory and Computation

Memory Saving As discussed in §3.1, we need to allocate memory for the KV cache budget $\boldsymbol{M}_{\text{budget}} \in \mathbb{R}^{b \times B_{\text{budget}} \times N_{\text{layer}} \times N_{\text{head}} \times d}$ to retain B_{budget} KV cache tokens, and for the buffer $\boldsymbol{M}_{\text{buffer}} \in \mathbb{R}^{b \times B_{\text{buffer}} \times N_{\text{layer}} \times N_{\text{head}} \times d}$ to store B_{buffer} newly generated KV cache tokens during the generation of a text segment. Here, b is the batch size, N_{layer} is the number of Transformer layers, N_{head} is the number of attention heads, and d is the dimension of attention heads. In addition, we also need to allocate memory for the model weight $\boldsymbol{M}_\theta$. During decoding, the previous query states are typically discarded by default, so we use a query cache to store the last α tokens in the query state, consuming memory of $\boldsymbol{M}_\alpha \in \mathbb{R}^{b \times \alpha \times N_{\text{layer}} \times N_{\text{head}} \times d}$. In summary, R-KV requires memory of $\boldsymbol{M}_{\text{total}} = \boldsymbol{M}_\theta + \boldsymbol{M}_{\text{budget}} + \boldsymbol{M}_{\text{buffer}} + \boldsymbol{M}_\alpha$ during generation. In comparison to FullKV without KV cache compression, generating B_{full} tokens requires memory of $\boldsymbol{M}_{\text{full}} \in \mathbb{R}^{b \times B_{\text{full}} \times N_{\text{layer}} \times N_{\text{head}} \times D_{\text{head}}}$ to retain B_{full} KV tokens, and memory of the model weight $\boldsymbol{M}_0$. Therefore, the memory saved by our method w.r.t. FullKV is: $\boldsymbol{M}_{\text{saving}} = \boldsymbol{M}_{\text{full}} - \boldsymbol{M}_{\text{budget}} - \boldsymbol{M}_{\text{buffer}} - \boldsymbol{M}_\alpha$.

Computation Overhead The computational complexity of importance scoring (See §3.2) is $O(\alpha B_{\text{budget}})$ while redundancy estimation (see §3.3) has complexity $O(B_{\text{budget}}^2)$. Thus, the total overhead incurred during each generation segment is $O(\alpha B_{\text{budget}} + B_{\text{budget}}^2)$. The generation complexity without KV cache compression is $O(B_{\text{full}} B_{\text{buffer}})$, whereas the complexity with KV cache compression is $O((B_{\text{budget}} + B_{\text{buffer}}) B_{\text{buffer}})$. For reasoning models, B_{full} tends to be large because of the long generation length, and using a relatively small B_{budget} value can efficiently reduce computation cost. The effectiveness of this approach depends on depends on whether the speedup gained by attending over a reduced KV cache outweighs the overhead of computing the compression scores—i.e., the combined cost of importance and redundancy scores, ($O(\alpha B_{\text{budget}}) + O(B_{\text{budget}}^2)$).

C.2 Detailed Analysis of Throughput Results

We analyze the end-to-end throughput from two perspectives: ratio budget and fixed budget.

Ratio Budget: section 4.2 indicates that for DeepSeek-R1-Distill-Llama-8B, lossless compression (i.e., model performance equivalent to no KV compression) is achievable when the KV budget ratio, relative to the output length, is between 10% and 34%. For DeepSeek-R1-Distill-Qwen-14B, this range for lossless compression is 25% to 54% of the output length. Consequentlywe investigated the maximum achievable batch size and corresponding throughput for R-KV at compression ratios of 10%, 34%, and 54%, comparing these against the maximum batch size and throughput of FullKV using DeepSeek-R1-Distill-Llama-8B. In 8K sequence length setting, at a 54% compression ratio, R-KV allows for a batch size 1.7 × larger than FullKV, resulting in 1.5 × the throughput. At a 10% compression ratio, R-KV achieves a 7.7 × increase in batch size and a 4.5 × increase in throughput compared to FullKV. For a 16K sequence length setting, at 54% compression, the batch size is 1.5 × that of FullKV, and the throughput is 1.7 × higher. At 10% compression, R-KV supports a 9 × larger batch size, delivering 6.6 × the throughput. We observe that for smaller batch sizes (e.g., less than 128), throughput scales nearly linearly with increasing batch size. However, for larger batch sizes this linear scaling diminishes as inference on the NVIDIA A100 GPU becomes compute-bound.

Fixed Budget: We also conducted an analysis under a fixed KV cache budget. With an output length of 8K and a fixed budget $B_{\text{budget}} = 1024$, R-KV enables a batch size 6.48 × larger than FullKV, yielding 3.8 × the throughput. At $B_{\text{budget}} = 1536$, the batch size is 4.6 × larger, and throughput is 3 × that of FullKV. For an output length of 16K and $B_{\text{budget}} = 1024$, R-KV achieves a 13.4 × increase in batch size and a 9.19 × increase in throughput. With $B_{\text{budget}} = 1536$, the batch size is 9.6 × larger, and throughput is 7.1 × higher. In the fixed budget scenario, the advantage of R-KV becomes more pronounced with longer generation lengths. This is because the KV cache size for R-KV under a fixed budget does not increase with the sequence length, unlike FullKV where the memory footprint grows linearly with the generation length, thus more severely limiting its maximum batch size.

Gen. Length	Method	Budget	Mem. Saving (%)	Batch	Throughput (tok/s)	Tokens Gen.	Dec. Time (s)
8K	FullKV	–	–	1	75.44	8 094	107.30
		–	–	62 (max)	849.13	501 828	590.99
	SnapKV	Fixed – 1024	87.50	1	81.26	8 094	99.60
		Fixed – 1024	87.50	402 (max)	3 253.93	3 253 788	999.96
		Fixed – 1536	81.25	287 (max)	2 525.25	2 322 978	919.90
		Fixed – 3072	62.50	150 (max)	1 527.67	1 214 100	794.74
		Ratio – 10% – 819	90.00	479 (max)	3 808.81	3 877 026	1 017.91
		Ratio – 34% – 2 785	66.00	167 (max)	1 625.46	1 351 698	831.58
		Ratio – 54% – 4 423	46.00	105 (max)	1 269.68	849 870	669.36
	R-KV	Fixed – 1024	87.50	1	80.46	8 094	100.60
		Fixed – 1024	87.50	402 (max)	3 251.52	3 253 788	1 000.70
		Fixed – 1536	81.25	287 (max)	2 525.75	6 546 972	919.72
		Fixed – 3072	62.50	150 (max)	1 520.99	1 214 100	798.23
		Ratio – 10% – 819	90.00	479 (max)	3 809.15	3 877 026	1 017.82
		Ratio – 34% – 2 785	66.00	167 (max)	1 608.01	1 351 698	840.61
		Ratio – 54% – 4 423	46.00	105 (max)	1 257.83	849 870	675.66
16K	FullKV	–	–	1	69.41	16 286	234.65
		–	–	30 (max)	347.03	488 580	1 407.89
	SnapKV	Fixed – 1024	87.50	1	81.03	16 286	200.99
		Fixed – 1024	87.50	402 (max)	3 202.17	6 546 972	2 044.54
		Fixed – 1536	81.25	287 (max)	2 449.02	4 674 082	1 908.56
		Fixed – 3072	81.25	150 (max)	1 413.84	2 442 900	1 727.84
		Ratio – 10% – 1 638	90.00	271 (max)	2 306.26	4 413 506	1 913.71
		Ratio – 34% – 5 570	66.00	82 (max)	798.42	1 335 452	1 672.61
		Ratio – 54% – 8 847	46.00	46 (max)	586.43	749 156	1 277.48
	R-KV	Fixed – 1024	93.75	1	80.95	16 286	201.18
		Fixed – 1024	93.75	402 (max)	3 188.82	6 546 972	2 053.10
		Fixed – 1536	90.63	287 (max)	2 447.61	4 674 082	1 909.65
		Fixed – 3072	81.25	150 (max)	1 406.28	2 442 900	1 737.13
		Ratio – 10% – 1 638	90.00	271 (max)	2 300.28	4 413 506	1 918.68
		Ratio – 34% – 5 570	66.00	82 (max)	797.43	1 335 452	1 674.70
		Ratio – 54% – 8 847	46.00	46 (max)	584.77	749 156	1 281.12

Table 3: Memory-saving, throughput, and decoding-time comparison for LLAMA3-8B under various generation lengths and KV-cache compression budgets.

C.3 Results

Full results could be found at Table 3. While `R-KV` incurs a minor computational overhead for redundancy estimation compared with SnapKV, this results in a throughput that is only slightly lower, with a negligible difference of less than 1%.

D Limitations

One limitation of our proposed KV cache compression method is its current compatibility with certain advanced attention mechanisms, such as paged attention. Adapting our compression technique to seamlessly integrate with such mechanisms presents a non-trivial challenge and may require further investigation. Additionally, the implementation of KV cache compression within existing serving frameworks can encounter practical difficulties, particularly if these frameworks lack native support or flexible interfaces for KV cache compression. In serving frameworks that do not offer specialized KV cache compression interfaces, the performance benefits of our method might be less pronounced. Without such interfaces, implementing KV cache compression may necessitate reallocating memory to store the compressed KV cache and subsequently deallocating the memory used for the original, uncompressed cache. This process of memory reallocation can introduce significant overhead, potentially offsetting some of the acceleration gains. In contrast, serving frameworks equipped with dedicated KV compression interfaces can handle these operations much more efficiently, avoiding such costly memory management tasks.

https://doi.org/10.52202/085713-2038

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: abstract and introduction

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitation

Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: 5.2 Failure of Attention-Based Methods to Capture Redundancy

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification:

   Guidelines:

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

https://doi.org/10.52202/085713-2038

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

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

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

   Answer: [Yes]

   Justification: code open-source

   Guidelines:

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

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: MATH-500 and AIME24

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Run experiments for 64 times and calculate averaged Pass@1.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: A100

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

https://doi.org/10.52202/085713-2038

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification:

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Conclusion: As a training-free and model-agnostic solution, R-KV offers a practical path to deploy advanced reasoning LLMs more efficiently and scalably.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

    Justification:

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification:

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification:

    Guidelines:

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

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

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

    Answer: [NA]

https://doi.org/10.52202/085713-2038

Justification:

Guidelines:

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

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

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

    Answer: [NA]

    Justification:

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification:

    Guidelines:

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