R-KVHASH: KV CACHE COMPRESSION OF REASONING TRACES VIA SIMHASH-BASED ESTIMATION OF REDUNDANT TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning models excel on benchmarks which benefit from multi-step reasoning. However, these reasoning traces are often excessively verbose, producing outputs that contain tens-of-thousands of tokens. The resulting key-value (KV) cache, which stores past token embeddings, grows linearly with sequence length. The R-KV compression algorithm addresses this issue by evicting tokens from the cache associated with redundant self-reflection and reasoning steps. However, this per-token redundancy estimation relies on calculating pairwise key cosine similarities, thus necessitating a Gram matrix product of the key cache, along with an accumulated attention score calculation. The associated memory and computational complexity of these operations become expensive with increasing budget and context size. We propose R-KVHash, which uses locality-sensitive hashing, namely the SimHash, to efficiently estimate the key similarities with sub-linear memory and computational complexity, and entirely avoids computation of attention-based importance. We find that our approach, which buckets keys through a binarized Gaussian projection, exhibits up to $2\times$ higher decoding throughput than R-KV and is also competitive in accuracy on MATH500 and GSM8K for DeepSeek-R1-Distill-Qwen 7B and 14B.

1 INTRODUCTION

Large language models (LLMs), which are trained to engage in extensive chain-of-thought demonstrate remarkable performance on benchmarks which require complex reasoning. However, this family of *reasoning models*, such as DeepSeek-R1 (Guo et al., 2025) and the NVIDIA Nemotron series (Adler et al., 2024), produce long outputs containing repetitive thinking sequences. The KV cache, which expedites attention calculations by maintaining the key and value embeddings of past tokens, scales linearly with the total size of context (i.e., tokens in both the prompt and response).

As an example, a popular open-source reasoning model, DeepSeek-R1-Distill-Qwen-7B, can generate responses up to nearly 16,0000 tokens long on a single question from the MATH-500 benchmark (Cai et al., 2025). The KV caches across all heads and layers for such a response would cost approximately ≈ 3.5GB to host. The required attention calculation for the next token would require a dot product of the current query against all cached keys, necessitating millions of floating-point operations.

A popular strategy for KV cache compression is to evict (i.e., delete) the key-value embeddings of past, "unimportant" tokens. Different heuristics exist for estimating importance, the most prominent being those which use accumulated attention scores (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024b) for each token. However, existing literature emphasizes the eviction of tokens during the pre-fill phase (the prompt), versus the autoregressive decoding phase (the response).

The **R**edundancy-Aware **KV** compression algorithm (R-KV) (Cai et al., 2025) is one of the only state-of-the-art compression strategies designed to compress the KV cache of reasoning models by exploiting the highly repetitive nature of reasoning traces (see Figure 2). However, to estimate redundant tokens, R-KV requires a Gram matrix product of the key cache to produce cosine similarities between all keys. For a cache budget of n tokens in $\mathbb{R}^d$, this similarity matrix costs $\mathcal{O}(n^2 d)$ to compute and $\mathcal{O}(n^2)$ host. That is, cost scales quadratically with budget size, which is often selected as a ratio of expected response length (Cai et al., 2025), thus potentially accumulating many tokens.

In this work, we propose R-KVHash, which uses locality-sensitive hashing to efficiently estimate these pairwise key similarities. In particular, we use the SimHash (Goemans & Williamson, 1995; Charikar, 2002), to project all keys to b-dimensional binary hash codes with $b \ll d$. This effectively buckets groups of keys together by hash code and uses the Hamming distances between bucket codes to estimate cosine similarity. The hash table over all keys costs $\mathcal{O}(bn)$ to host and $\mathcal{O}(bn^2)$ to compute all distances, significantly improving inference and memory efficiency. We further demonstrate that this LSH-based similarity estimation is a sufficient importance heuristic on its own and avoids the need for calculation of attention-based importance. In particular, for a zero-shot pass@1 evaluation of the GSM8K and MATH500 benchmarks, we consistently observe performance gains over R-KV. This indicates that LSH is highly effective in grouping (un)important reasoning tokens.

2 Preliminaries

In this section, we detail the redundancy-aware R-KV algorithm and our SimHash variant, R-KVHash. We consider a decoder-only transformer with multi-head attention (Vaswani et al., 2017). For a sequence of n tokens, we denote the query cache $Q \in \mathbb{R}^{n \times d}$, key cache $K \in \mathbb{R}^{n \times d}$, and value cache $V \in \mathbb{R}^{n \times d}$, where d is the hidden dimension.

2.1 R-KV Compression

To determine which tokens to evict from the cache, the R-KV algorithm assigns an *attention-based importance score* and *redundancy score*, which are jointly used to determine which tokens to evict and retain in the cache. As depicted in Figure 2, various reasoning and self-reflective steps are frequently repeated, thus motivating the eviction of redundant tokens.

2.1.1 Attention-based Importance

Following the SnapKV (Li et al., 2024b) framework, to estimate the attention-based importance for a token, R-KV considers the attention distributions of the last α tokens. For each attention head h, the attention matrix $A^h \in \mathbb{R}^{\alpha \times d}$ is calculated as,

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

where $Q^h \in \mathbb{R}^{\alpha \times d}$ is the query cache for head h restricted to the last α tokens and $K^h \in \mathbb{R}^{n \times d}$. For grouped-query attention (GQA) (Ainslie et al., 2023), key/values are shared among G groups of heads (typically $G = 2, 4, 8$), thus, as an abuse of notation we denote K^h and V^h as the shared value caches, and $Q^{h,0}, \ldots, Q^{h,G-1} \in \mathbb{R}^{\alpha \times d}$ as the query caches of each head h by group. For GQA, we first compute the group attention matrix over our observation window as

$$A^{h,g}_{\text{group}} = Q^{h,g} \cdot (K^h)^\top / \sqrt{d},\ 0 \leq g \leq G-1. \tag{2}$$

We then calculate a final group attention score as

$$A^h_{\text{group}} = \text{maxpool}\left(\left[A^{h,0}_{\text{group}}, \ldots, A^{h,G-1}_{\text{group}}\right]\right), \qquad A^h = \text{softmax}\left(A^h_{\text{group}}\right). \tag{3}$$

The entry $A^h_{i,j}$ denotes the attention from the j-th query token to the i-th key in head h. Following Li et al. (2024b), we define the attention-based importance score I^h_i of the i-th token in head h as

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

2.1.2 Redundancy Estimation

R-KV attempts to estimate redundancy by calculating the cosine similarity among key tokens. Specifically, for a key cache $K^h \in \mathbb{R}^{n \times d}$ in head h, the cosine similarity matrix S^h is calculated as

$$S^h = \overline{K}^h (\overline{K}^h)^\top \in \mathbb{R}^{n \times n}, \tag{5}$$

where $K^h_i = \frac{K^h_i}{\|K^h_i\|_2}$ and $S^h_{i,i} \leftarrow 0, \forall i \in [0, n)$ to avoid considering redundancy of a key against itself. Furthermore, for row i, we filter for similarity scores above a user-defined threshold T and zero out

entries corresponding to the β most recent tokens to eliminate them from consideration for eviction. Maintenance of a recency window is common to many KV compression algorithms (Zhang et al., 2023; Li et al., 2024b; Liu et al., 2023), as newly-generated tokens are known to be important for future decoding steps. Now, we will compute an average similarity score $\bar{S}_i^h$ and overall redundancy score $R_i^h = \text{softmax}\left([\bar{S}_0^h, \ldots, \bar{S}_{n-1}^h]\right)_i$, where $\bar{S}_i^h = \frac{1}{n}\sum_{j=0}^{n-1} S_{j,i}^h$.

At a high level, $S_{j,i}^h$ estimates how similar the i-th key is to all other tokens in the cache of head h, which is then transformed into a per-token redundancy distribution R_i^h, with higher scores indicating greater redundancy. R-KV then utilizes a joint selection score: $Z_i^h = \lambda I_i^h - (1-\lambda)R_i^h$, where λ is a tunable hyperparameter to balance emphasis between attention and redundancy-based importance. A buffer β of recent tokens is used to determine the cache eviction rate. Every β decoding steps, Z_i^h is calculated for all tokens currently in the cache, and those with the top B scores are kept.

2.2 R-KVHash

Our core contribution, ***R-KVHash***, replaces R-KV's pairwise cosine-similarity computation with a locality-sensitive hashing (LSH) based redundancy estimator, avoiding computation of attention-based importance scores. Computing cosine similarity across all token pairs requires $\mathcal{O}_n(n^2)$ operations for n tokens, significantly slowing inference. R-KVHash reduces this to $\mathcal{O}_n(n)$ complexity.

We adopt a projection-based hash function, $\overline{K'}^h = \overline{K}^h R$, where $\overline{K}^h \in \mathbb{R}^{n\times d}$ is the normalized key cache for head h, and $R \in \mathbb{R}^{d\times b}$ is a random Gaussian matrix sampled once before inference. Each row of $\overline{K'}^h$ is converted into a b-bit binary code by thresholding its entries at zero (positive $\rightarrow 1$, negative $\rightarrow 0$). This assigns each key to one of 2^b hash buckets. We denote this mapping by $H(\cdot)$. This strategy, known as the SimHash (Charikar, 2002; Goemans & Williamson, 1995), enjoys the following property for $x, y \in \mathbb{R}^d$:

$$\frac{1}{b} \cdot \mathbb{E}[\text{Hamming}(H(x), H(y))] = \arccos(\theta), \tag{6}$$

where $\text{Hamming}(\cdot,\cdot)$ denotes the Hamming distance and expectation is with respect to the randomness of the Gaussian matrix. Therefore, we may use this as an estimator of cosine similarity between keys.

Let c_i denote the number of keys in bucket i. The redundancy score for bucket i is defined as

$$S'_i = \frac{\sum_{i\neq j} c_j \, \cos(\text{Hamming}(i,j)/b)}{\sum_j c_j}, \tag{7}$$

where $\text{Hamming}(i,j)$ is the precomputed Hamming distance between bucket indices i and j. For a specific key i, the redundancy score is $S_i = S'_{H(\overline{K'}_i^h)}$. Every token i in the cache is ranked by their score S_i, and the lowest-scoring tokens are dropped to meet the specified cache budget.

In R-KVHash, each key requires only a single matrix projection ($\mathcal{O}(1)$ to host). All Hamming distances can be precomputed offline in $\mathcal{O}(n^2 b)$. Thus, we reduce redundancy estimation from $\mathcal{O}_n(n^2)$ to $\mathcal{O}_n(n)$ overall. At a high-level, R-KVHash is approximating similarity in high-dimensional space by projecting keys into a lower-dimensional binary space, where proximity is measured by bucket membership. Keys assigned to the same or nearby buckets are therefore statistically more likely to represent redundant information. Therefore, R-KVHash can identify redundant keys using bucket similarity, effectively mirroring the intuition behind cosine similarity–based redundancy detection, while reducing computational and memory costs.

Bucket size b controls hashing granularity. Larger b produces more buckets and distributes keys more uniformly, but risks overly random evictions. Smaller b aggregates many keys into the same bucket, potentially discarding important tokens. Through fine-tuning, we found $b = 16$ most performant.

3 Experiments

3.1 Setup

Models and Datasets. We test two DeepSeek-R1 distillations, R1-Distill-Qwen-7B, and a larger R1-Distill-Qwen-14B. The performance of these models is assessed on the MATH-500 benchmark

Method	R1-Distill-Qwen-7B		R1-Distill-Qwen-14B	
	MATH-500	GSM8K	MATH-500	GSM8K
Full	0.63	0.50	0.64	0.82
R-KV	0.23	0.43	0.22	0.57
R-KVHash	0.41	0.45	0.40	0.75

Table 1: **R-KV vs. R-KVHash on MATH-500 and GSM8K.** Q-7B and Q-14B denote R1-Distill-Qwen-7B/14B. Each model is allowed a budget of 1,024 tokens.

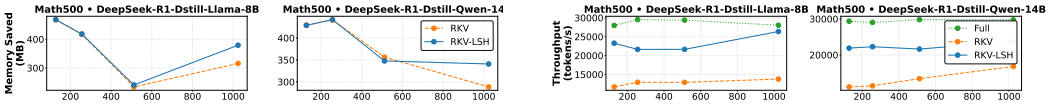


Figure 1: Memory saved (MB) and decoding-phase throughput by R-KV and R-KVHash, with max token length set to 16,384. R-KVHash achieves greater memory savings as cache budget increases and substantially higher throughput. Averages over 50 samples are reported.

(Hendrycks et al., 2021), consisting of 500 competition math problems and the train split of GSM8K, containing 7k grade-school math problems, from which we sample 500 questions.

Hyperparameters. For both R-KV and R-KVHash, we set the buffer size $\beta = 128$ and window size $\alpha = 8$, mimicking the settings of Cai et al. (2025). We set the recommended $\lambda = 0.1$ for R-KV. We draw a single random Gaussian for R-KVhash and fix it as the projection matrix for all experiments using `numpy` seed=42. Following Cai et al. (2025), on MATH-500, we set a max-token generation length of 16,384. For GSM8K, we set a max-token generation length of 8,000.

3.2 Results

In Table 1, we report pass@1 accuracies for the full KV cache, R-KV, and R-KVHash on MATH-500 for R1-Llama-7B and R1-Qwen-14B. R-KVHash outperforms R-KV and is much closer to full-model accuracy, indicating the utility of our LSH-based redundancy score.

Figures 1 compares memory consumption and decoding throughput between R-KV and R-KVHash for an alternative model DeepSeek-R1-Distill-Llama-8B and DeepSeek-R1-Distill-Qwen-14B on MATH500 which elicits longer responses on average. We first note that the storage costs for the similarity calculation are markedly reduced. For R-KVHash, our similarity tables use kilobytes worth of overhead as they are completely maintained in binary. This is in contrast to R-KV, which uses full FP64 precision, which would be even more expensive for architectures such as the Llama-3 series (Dubey et al., 2024) which uses deeper embedding dimensions.

The decoding throughput of R-KVHash is nearly $2\times$ that of R-KV due to the efficiency of our scoring function (Equation 7), which uses LSH-based similarity and does not compute accumulated attention. For the R1-Distill-Llama-8B architecture, the throughput of R-KVHash matches that of uncompressed inference. While R-KV relies on an exact average of cosine similarity to determine overall redundancy along with accumulated attention scores, we find that grouping tokens by low-dimensional hash codes functions as a superior strategy for bucketing irrelevant tokens.

4 Conclusion

Reasoning models produce long outputs resulting in prohibitively large KV caches. R-KV is a notable effort to compress reasoning KV caches by targeting repetitive thinking tokens via a joint cosine-similarity-based redundancy and attention-importance score. However, this incurs memory and computational costs which scale quadratically with budget. We introduce a SimHash-based variant, R-KVHash, which efficiently buckets similar tokens. We find in our experiments that R-KVHash is significantly more efficient and accurate than R-KV.

REFERENCES

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, et al. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*, 2024.

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Yash Akhauri, Ahmed F AbouElhamayed, Yifei Gao, Chi-Chih Chang, Nilesh Jain, and Mohamed S. Abdelfattah. Tokenbutler: Token importance is predictable. *arXiv preprint arXiv:2503.07518*, 2025.

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

Ngoc Bui, Shubham Sharma, Simran Lamba, Saumitra Mishra, and Rex Ying. Cache what lasts: Token retention for memory-bounded kv cache in llms. *arXiv preprint arXiv:2512.03324*, 2025.

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

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, et al. R-kv: Redundancy-aware kv cache compression for training-free reasoning models acceleration. *arXiv preprint arXiv:2505.24133*, 2025.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 380–388, 2002.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. Magicpig: Lsh sampling for efficient llm generation. *arXiv preprint arXiv:2410.16179*, 2024.

MAA Codeforces. American invitational mathematics examination-aime 2024, 2024.

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

Shaoting Feng, Yuhan Liu, Hanchen Li, Xiaokun Chen, Samuel Shen, Kuntai Du, Zhuohan Gu, Rui Zhang, Yuyang Huang, Yihua Cheng, Jiayi Yao, Qizheng Zhang, Ganesh Ananthanarayanan, and Junchen Jiang. Evicpress: Joint kv-cache compression and eviction for efficient llm serving. *arXiv preprint arXiv:2512.14946*, 2025a.

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.

Yuan Feng, Haoyu Guo, JunLin Lv, S. Kevin Zhou, and Xike Xie. Taming the fragility of kv cache eviction in llm inference. *arXiv preprint arXiv:2510.13334*, 2025b.

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.

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6): 1115–1145, 1995.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters. *arXiv preprint arXiv:2406.12335*, 2024.

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.

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

Yen-Chieh Huang, Pi-Cheng Hsiu, Rui Fang, and Ming-Syan Chen. Kv admission: Learning what to write for efficient long-context inference. *arXiv preprint arXiv:2512.17452*, 2025.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Mo Li, Songyang Zhang, Taolin Zhang, Haodong Duan, Yunxin Liu, and Kai Chen. Needlebench: Evaluating llm retrieval and reasoning across varying information densities. *arXiv preprint arXiv:2407.11963*, 2024a.

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

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

Minghui Liu, Aadi Palnitkar, Tahseen Rabbani, Hyunwoo Jae, Kyle Rui Sang, Dixi Yao, Shayan Shabihi, Fuheng Zhao, Tian Li, Ce Zhang, Furong Huang, and Kunpeng Zhang. Hold onto that thought: Assessing kv cache compression on reasoning. *arXiv preprint arXiv:2512.12008*, 2025a.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. *arXiv preprint arXiv:2502.00299*, 2025b.

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

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

Akshat Ramachandran, Marina Neseem, Charbel Sakr, Rangharajan Venkatesan, Brucek Khailany, and Tushar Krishna. Thinkv: Thought-adaptive kv cache compression for efficient reasoning models. *arXiv preprint arXiv:2510.01290*, 2025.

Yiqun Shen, Song Yuan, Zhengze Zhang, Xiaoliang Wang, Daxin Jiang, and Cam-Tu Nguyen. LAVa: Layer-wise KV cache eviction with dynamic budget allocation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13672–13692, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.737. URL https://aclanthology.org/2025.findings-emnlp.737/.

Yuhang Shi, Bozhong Tian, Bofei Gao, Chen Li, Yucheng Li, Tianyu Liu, Junjie Hu, Zhiyuan Liu, and Maosong Sun. Pyramidinfer: Pyramid-based inference for long-context large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. If you prefer the ACL Anthology BibTeX (with pages/DOI), swap this stub for the official entry.

Brady Steele. On the limits of learned importance scoring for kv cache compression. *arXiv preprint arXiv:2601.14279*, 2026.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zihao Wang, Bin Cui, and Shaoduo Gan. Squeezeattention: 2d management of kv-cache in llm inference via layer-wise optimal budget. *arXiv preprint arXiv:2404.04793*, 2024.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. Longproc: Benchmarking long-context language models on long procedural generation. *arXiv preprint arXiv:2501.05414*, 2025.

Amir Zandieh, Insu Han, Vahab Mirrokni, and Amin Karbasi. Subgen: Token generation in sublinear time and memory. *arXiv preprint arXiv:2402.06082*, 2024.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. $infty$Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.814. URL `https://aclanthology.org/2024.acl-long.814/`.

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

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. DynamicKV: Task-aware adaptive KV cache compression for long context LLMs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8042–8057, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.426. URL `https://aclanthology.org/2025.findings-emnlp.426/`.

APPENDIX

A LLM USAGE

This work used LLMs solely to polish grammar and underlying LaTeX code. LLMs were neither used for writing style nor content suggestions.

B RELATED WORKS

R-KVHash falls into the family of methods that compress KV caches via token eviction and selection. Token eviction relies on a metric to determine past tokens that are important for the current decoding

step. Popular methods such as H2O (Zhang et al., 2023), SnapKV (Li et al., 2024b), and Scissorhands (Liu et al., 2023) use accumulated attention statistics of past tokens (often measured against a window of recent tokens). FastGen (Ge et al., 2023) profiles head-specific preferred token types. Recent work expands this space via (i) richer token-importance indicators that incorporate values (Guo et al., 2024), (ii) layer-/budget-aware cache management (Cai et al., 2024; Shi et al., 2024; Wang et al., 2024; Feng et al., 2024; Shen et al., 2025; Zhou et al., 2025; Qin et al., 2025), and (iii) query-aware retention mechanisms (Tang et al., 2024; Akhauri et al., 2025). While many methods were initially evaluated primarily on long-prompt understanding benchmarks such as LongBench (Bai et al., 2023) and RULER (Hsieh et al., 2024), newer evaluations also include more extreme controlled settings such as ∞Bench (Zhang et al., 2024) and NeedleBench (Li et al., 2024a), as well as long-output procedural generation suites (Ye et al., 2025).

The R-KV algorithm (Cai et al., 2025) uses a dual attention-based and redundancy score to assess token importance. Our method, R-KVHash, alters the redundancy calculation by leveraging locality-sensitive SimHash to bucket highly cosine-similar keys together, enabling a cheaper approximation to between-key similarity. Several recent works also emphasize robustness in decoding-time eviction and reasoning-centric regimes, including ThinKV (Ramachandran et al., 2025), DefensiveKV (Feng et al., 2025b), and joint compression–eviction approaches (Feng et al., 2025a). Orthogonal lines of work explore admission/retention at write time, e.g., semantic chunk-based compression (Liu et al., 2025b), learned token retention (Bui et al., 2025), and write-gated admission (Huang et al., 2025). Recent analyses further highlight that eviction behavior can differ substantially between prompt-compression settings and long-reasoning decodes (Liu et al., 2025a), and study limits of learned importance scoring (Steele, 2026).

R-KVHash is also related to hashing- and LSH-based acceleration of attention. HashEvict (Liu et al., 2024) uses SimHash on prior keys and the current query to target cosine-dissimilar tokens for eviction, though not to estimate between-key similarity as in R-KV and R-KVHash. Reformer (Kitaev et al., 2020), SubGen (Zandieh et al., 2024), and MagicPIG (Chen et al., 2024) use SimHash/LSH codes to cluster keys and sample representatives to approximate and/or speed up attention rather than evict tokens. In contrast, R-KVHash uses SimHash explicitly as a redundancy-detection primitive for eviction, offering complementary efficiency gains to methods that focus on allocation, quantization, or learned admission/retention.

R-KVHash is also related to hashing- and LSH-based acceleration of attention. HashEvict (Liu et al., 2024) applies SimHash to prior keys and the current query to target cosine-dissimilar tokens for eviction, but does not estimate between-key similarity as in R-KV and R-KVHash. Reformer (Kitaev et al., 2020) and subsequent SimHash/LSH-based approximations such as SubGen (Zandieh et al., 2024) and MagicPIG (Chen et al., 2024) use hashing to cluster keys and sample representatives to approximate and/or speed up attention, rather than to evict tokens. In contrast, R-KVHash uses SimHash explicitly as a redundancy-detection primitive for eviction, offering complementary efficiency gains to methods that focus on allocation, quantization, or learned admission/retention.

C Limitations

Evaluating a pass@1 for a higher number of responses may help determine how many responses are needed to achieve maximal accuracy. Furthermore, this would allow experiments with more difficult datasets such as AIME24 and AIME25 (Codeforces). In Cai et al. (2025) a total of 64 responses were allowed per question while we opt for a single response to mimic the resource-constrained scenario. Additionally, replacing the attention-importance function via an LSH akin to HashEvict (Liu et al., 2024) would further reduce efficiency costs, though we do not explore this.

While we do not anticipate any significant societal impacts of this work, it must also be noted that any compression strategy alters the LLM response, thus such strategies should be carefully considered in domains where any deviation from the vanilla output is undesirable.

D Redundant Reasoning

Reasoning models are notorious for repetitive thinking sequences, which we depict in Figure 2.

Q: Let $P(x)$ be a monic polynomial of degree 3. Suppose that $P(x)$ has a remainder of $R(x)$ when it is divided by [....] Find $P(5)$.

A: 250/2 -125/2 -105/2 +10/2 = (250 -125 -105 +10)/2 = (250 -230)/2 = 20/2 =10. Wait, that's 10? Wait, that contradicts the previous result. Hmm, something's wrong here. Wait, let me recalculate: 125 is 125 Minus 125/2 is 62.5 Minus 105/2 is 52.5 Plus 5 is 5 So, 125 -62.5 -52.5 +5 = 125 -115 +5 = 15. But when I did it with fractions, I got 10. Hmm, must have messed up the fraction conversion. [...]

Figure 2: A response from DeepSeek-R1-Distill-Qwen-7B to a MATH-500 question with repetitive sequences.

E TOKEN IMPORTANCE HEATMAPS

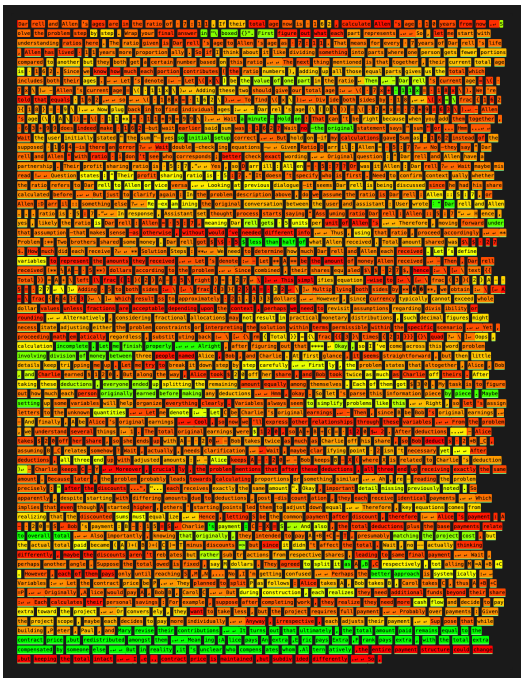
These heatmaps visualize the importance scores assigned to each token during KV cache compression with RKV and RKVHash. Each colored cell represents a token from the input question and model-generated response.

Color encoding:

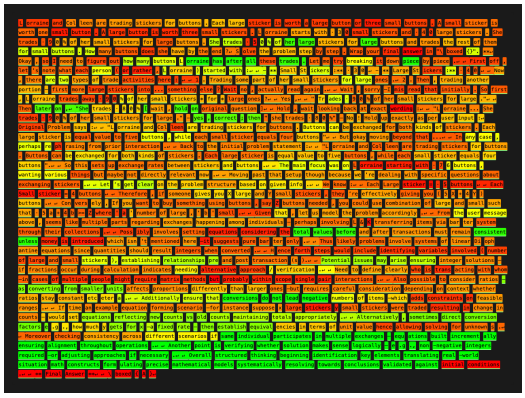
- Red: Low importance (likely to be evicted from cache)
- Yellow: Medium importance
- Green: High importance (retained in cache)

The importance scores are computed as the sum of attention weights across all heads at each compression step. Scores are normalized using percentile ranking for visualization clarity.

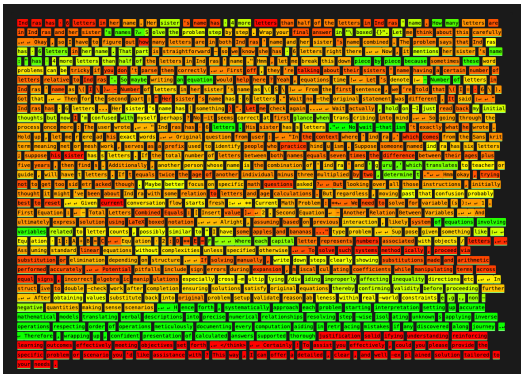
Key observations: Input tokens (question) often receive lower importance scores than recently generated tokens. This recency bias can cause critical information (numbers, names) to be evicted. We find in Figure 6 that R-KV demonstrates recency bias. R-KVHash incorporates LSH-based redundancy penalties, potentially preserving more diverse tokens as demonstrated in Figure 7.



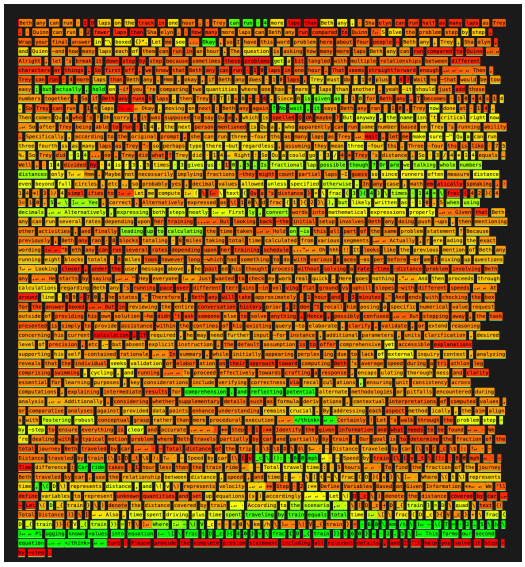
(a) Question 0



(b) Question 1

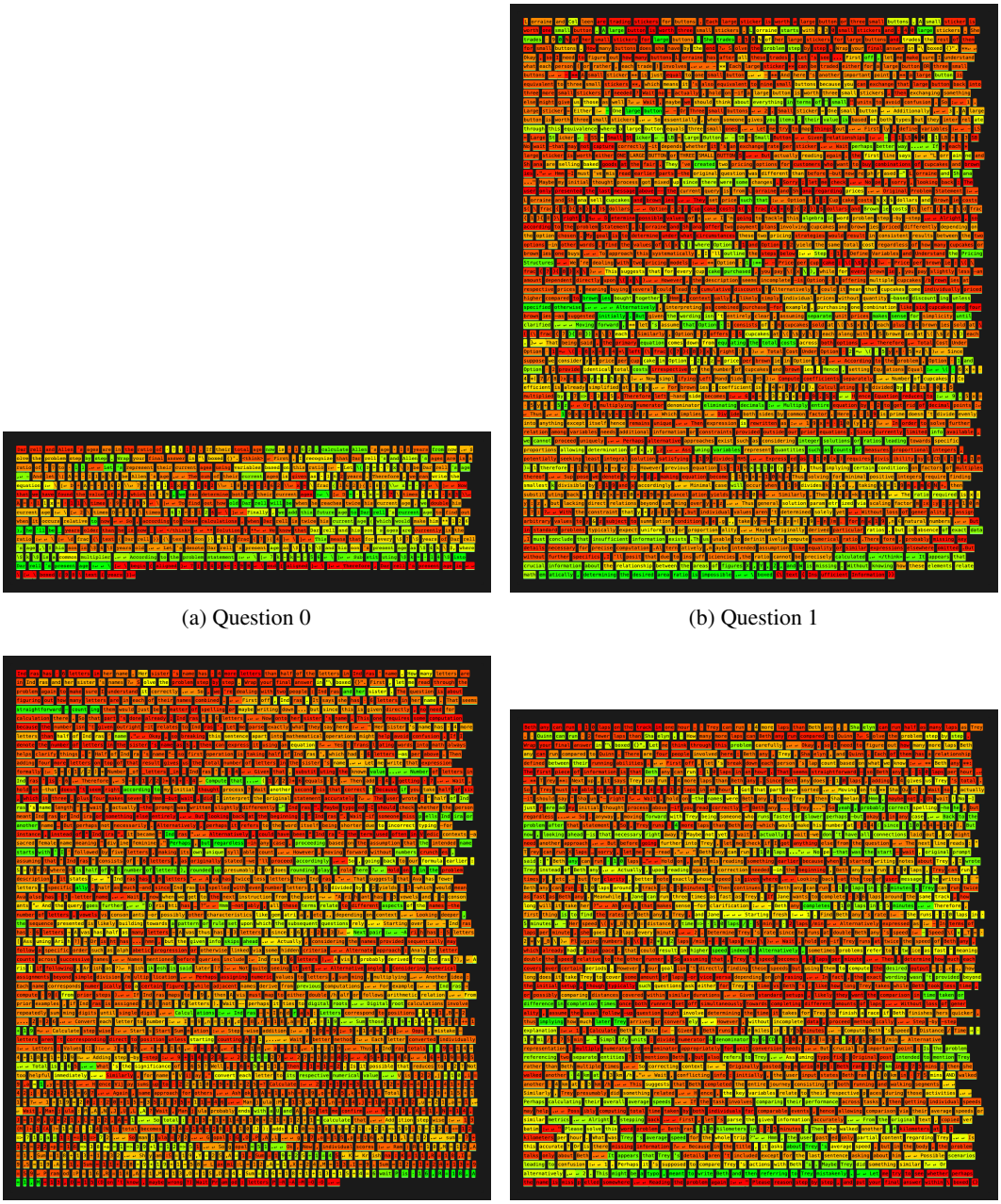


(c) Question 2



(d) Question 3

Figure 3: **Token Importance Heatmaps for R-KV.** Four example questions (top-left to top-right, then bottom-left to bottom-right: Questions 0–3) from R1-Distill-Qwen-7B with budget=128. Each colored cell represents a token from the input question and model-generated response. *Red* indicates low importance (likely to be evicted from cache), *yellow* indicates medium importance, and *green* indicates high importance (retained in cache). Importance scores are computed as the sum of attention weights across all heads at each compression step.



(a) Question 0

(b) Question 1

(c) Question 2

(d) Question 3

Figure 4: **Token Importance Heatmaps for R-KVHash.** Four example questions (top-left to top-right, then bottom-left to bottom-right: Questions 0–3) from R1-Distill-Qwen-7B with budget=128. R-KVHash incorporates LSH-based redundancy penalties to identify and evict redundant tokens, potentially preserving more diverse tokens compared to standard R-KV. Color encoding matches Figure 3.