ReFreeKV: Towards *Threshold-Free* KV Cache Compression

Xuanfan Ni[*♣] **Liyan Xu**[*†♥] **Chenyang Lyu**[♦] **Longyue Wang**[♦]
Mo Yu[♥] **Lemao Liu**[♥] **Fandong Meng**[♥] **Jie Zhou**[♥] **Piji Li**[♣]
[♣]Nanjing University of Aeronautics and Astronautics
[♥]WeChat AI, Tencent [♦]Independent Researcher
xuanfanni@gmail.com liyanlxu@tencent.com

Abstract

To reduce memory consumption during LLM inference, prior works have proposed numerous methods that focus on KV cache pruning based on various designed criteria. While these techniques often accomplish lossless memory reduction on many datasets, they often hinge on an under-emphasized condition: an input/domain-specific budget threshold needs to be pre-determined to achieve the optimal performance. However, such input-sensitive design may be considerably limited in real-world scenarios, as open-domain inputs span diverse domains, lengths and difficulty levels, without clear boundaries for threshold selection. As a result, the dependence of such input-sensitive threshold can be a fundamental limitation that causes large degradation on arbitrary inputs.

In this work, we propose a new objective that lifts the threshold constraints for robust KV compression, advocating for "*threshold-free*" methods that adaptively adjust budget allocation while preserving full-cache performance. We then propose a novel method, ReFreeKV, serving as the first instantiation of this objective. Extensive experiments across 13 datasets with diverse context lengths, task types, and model sizes demonstrate its efficacy and efficiency.

1 Introduction

Transformer-based large language models (LLMs) generate text autoregressively and maintain a *KV Cache* to store intermediate states during inference (Chang et al., 2023; OpenAI, 2023; Minaee et al., 2024). At each decoding step, the model retrieves the cached key and value vectors of all previous tokens, which typically reside in GPU memory for attention computation (Vaswani et al., 2017; Shazeer, 2019; Ainslie et al., 2023). Consequently, managing the KV cache efficiently has become crucial to mitigate the overall memory consumption and inference overhead, as they grow proportionally with the model size and the input length. For instance, Llama3-8B (Dubey et al., 2024) requires 1GB of KV cache memory for a 2K-token input, while its 70B counterpart demands a gigantic memory up to 50GB for 20K tokens.

Towards the KV cache efficiency, numerous recent methods have been proposed to effectively reduce the KV footprint after LLM prefilling. Exploiting the *sparsity* of attention, prior works have demonstrated that retaining full KV cache is not always necessary. Several optimization methods, such as H2O (Zhang et al., 2023), ScissorHands (Liu et al., 2023), SnapKV (Li et al., 2024), FastGen (Ge et al., 2024), CAKE (Qin et al., 2025), etc., discard the less critical cache positions according to their designed pruning criteria. Other paradigms such as KVMerger (Wang et al., 2024) and D2O (Wan et al., 2024) resort to merge or compress KV vectors instead of hard-pruning for achieving the memory reduction effects.

However, almost all prior KV reduction techniques hinge on a fundamental yet often underemphasized condition: a **data-dependent budget threshold** is typically involved to selectively tune for satisfactory results. For instance, D2O concludes a pre-defined KV cache budget ratio of 20% to match full-cache performance on LongBench (Bai et al., 2024), whereas our experiments suggest that the required budget can rise to 80% on GSM8K to maintain full performance. Similarly, retaining 1024 cache positions is sufficient for CAKE on LongBench, yet the same budget underperforms on the needle-in-the-haystack test (Kamradt, 2023).

The existence of such threshold serves fine in idealized *research settings* given dedicated datasets. Yet, its applicability may be considerably limited in *real-world scenarios*, where the ***inputs are intermixed across different domains, lengths and difficulty levels without explicit separation***. As a

[*] Equal contribution. Partial work done during Xuanfan's internship at Tencent. Correspondence to: Liyan Xu, Piji Li.
[†] Project lead: Liyan Xu <liyanlxu@tencent.com>.

Findings of the Association for Computational Linguistics: ACL 2026, pages 30827–30841

	GSM8K	GPQA	CoQA	Avg.
Budget = 50%				
H20	41.29%	9.99%	99.30%	50.19%
SLM	4.53%	20.78%	84.72%	36.68%
SnapKV	17.22%	26.91%	100.19%	48.11%
Budget = 20%				
H2O	3.23%	8.48%	96.53%	36.08%
SLM	4.26%	11.54%	76.05%	30.62%
SnapKV	1.71%	3.86%	98.60%	34.72%

Table 1: KV pruning methods that depend on a preset KV budget threshold can exhibit inconsistent performance across domains (percentage relative to full-cache scores using Llama3), making input-specific threshold selection unavoidable for achieving optimal inference.

result, optimal thresholds cannot be pre-determined for diverse real-world inputs, making the system less robust and prone to significant performance degradation in practice.

To further illustrate, Table 1 presents preliminary experiments using H2O, StreamingLLM (Xiao et al., 2024) and SnapKV, where inputs from different datasets are mixed to demonstrate the drawback of data-specific thresholds: *a threshold reaching full performance on one dataset may not transfer well to others*. As the KV cache budget ratio changes, their performance varies substantially across datasets.

Such inconsistency on the performance motivates us to revisit the goal of KV cache pruning. Rather than targeting only how much memory can be saved on a fixed benchmark, we instead ask for a **new objective**: *lifting the dependency of data-specific thresholds in KV cache pruning*, such that the system should robustly handle arbitrary inputs with *consistent* full-cache performance.

Specifically, our objective prioritizes two principles: **1)** the method may operate with a universal threshold insensitive to inputs, effectively rendering it "*threshold-free*"; and **2)** the method should consistently achieve performance comparable to its full-cache counterpart, able to dynamically adjust KV cache budgets. Pursuing the best possible compression ratio only comes after satisfying these two criteria. As shown by our full experimental results in Table 2, while prior methods may achieve strong compression on certain datasets, none could fulfill our objective to achieve consistent performance across diverse inputs.

Towards this objective, we propose **ReFreeKV**, a novel method featuring thReshold-Free KV cache pruning. ReFreeKV adopts a two-stage process with an input-insensitive threshold metric to dynamically control the KV cache budget. Conceptually, it first ranks all KV cache positions based on their positional importance; then, it progressively retains key-value vectors in order, and discards the remaining KV cache once the stopping criterion is met. For minimal overhead, ReFreeKV is designed for implementation with parallel operators, instead of sequential processing. As shown in Section 4.3, its latency is on par with prior efficient KV pruning methods across varying batch sizes.

The *threshold-free* aspect stems from a universal metric in the stopping criterion, termed *Uni-Metric*, whose design is shown to be consistent and **insensitive to variations** in input domains and sequence lengths. It should be noted that *threshold-free* does not mean the absence of any thresholding mechanism; though, we intentionally use this framing to emphasize that the efficacy of such method does not rely on a pre-determined threshold, thus no threshold calibration is needed in practice. By design, ReFreeKV naturally adjusts to higher compression ratios on simpler tasks, while allocating more cache resources to more complex ones.

To evaluate ReFreeKV, our experiments adopt 13 datasets varying diverse context lengths and tasks, including mathematical and commonsense reasoning, reading comprehension and coding, which show that ReFreeKV accomplishes our proposed objective. The resulting inference is highly comparable and can even surpass the full-cache performance, evaluated with multiple LLMs of different sizes, all without an input-specific threshold. For instance, using Llama3-8B, it automatically allocates an average KV budget ratio of 63.7%, while slightly exceeding full-cache performance across 13 datasets on average. Though the best compression ratios on a few datasets are not achieved by ReFreeKV, however, it remains the only method that could maintain decent compression under the *consistent performance constraint*, successfully addressing the limitation of existing baselines requiring data-specific budget thresholds.

2 Related Work

KV Cache Pruning To mitigate the large memory footprint of the KV cache during LLM inference, a prominent line of work has focused on pruning, which selectively discards less important cache positions. Methods like Scissorhands (Liu et al., 2023) and SnapKV (Li et al., 2024) retain to-

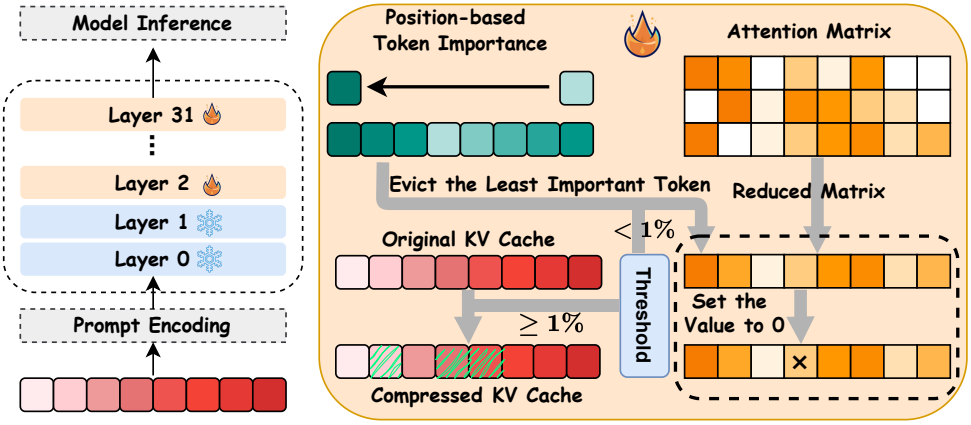

Figure 1: The overall workflow of ReFreeKV in Section 3.2. After prefilling, tokens are initially ranked based on their positions, followed by the eviction of the least significant tokens (per layer), whose halting condition is determined by the norm value of the reduced attention matrix. The KV cache for the remaining tokens are then preserved to subsequent generation.

kens based on high attention scores. Others employ strategies based on token recency and historical importance, such as H2O (Zhang et al., 2023) or StreamingLLM (Xiao et al., 2024). FastGen (Ge et al., 2024) further refines this by adapting retention strategies on a per-head basis.

Input-Sensitive Design in Recent Works While the shift from fixed-budget constraints to heuristic-based halting conditions is an established paradigm in recent literature, existing approaches often remain constrained by input-sensitive hyperparameters. Lethe (Zeng et al., 2026) introduces layer- and time-adaptive pruning during decoding, but still relies on *sparse* and *recent* ratio without particular emphasis on cross-task stability. SABlock (Chen et al., 2025) employs semantic-aware token scoring with adaptive block sizes, yet requires a predefined cache budget tuned per task. DuoAttention (Xiao et al., 2025) classifies heads into retrieval and streaming types with a constant-length cache for streaming heads, but this fixed window size is not designed for varying input complexity. AdaKV (Feng et al., 2024) proposes head-wise adaptive budget allocation guided by a theoretical loss bound, yet it essentially redistributes a pre-defined global budget across heads rather than eliminating the budget constraint itself. Although Twilight (Lin et al., 2025) removes the explicit budget via a top-p inspired mechanism, it shifts the burden to tuning the p value across models and inputs. In contrast, ReFreeKV aims to eliminates task-dependent threshold search; through validation across diverse datasets and models, it preserves full-cache performance robustly without any manual tuning.

3 Methodology

In this section, we first elaborate our motivation and the new objective for KV cache compression differing from prior works. We then delineate our proposed approach ReFreeKV, along with its key implementation details.

3.1 *Threshold-Free*: A New Objective

As discussed in Section 2, prior methods of KV cache compression require a pre-selected budget threshold in various forms. While these methods can perform well on numerous datasets, it is inevitable that such dependence on a threshold can become a practical limitation. As the optimal threshold can vary across different inputs, it is not always feasible for such systems to pick appropriate thresholds in maintaining stable performance involving arbitrary inputs and open-domain instructions. Certain inputs may necessitate a relatively higher memory budget, such as tasks with multi-step or mathematical reasoning, whereas others such as straightforward QA queries need only a small set of KV cache. Inputs in real-world scenarios, especially, are intermixed and unpredictable, with no clear boundary by difficulty or domain.

Our propose objective is exactly to remove input-specific threshold constraints, motivating *threshold-free* methods that ensure consistent performance comparable to full-cache regardless of inputs, while obviating the need of tuning for optimal thresholds. Pursuit of the best compression ratio is prioritized only after satisfying this aspect. To the best of our knowledge, we are the first to propose an effective solution that fulfills such objective.

3.2 ReFreeKV

ReFreeKV consists of two stages implemented with efficient parallel operators. Conceptually, it first ranks all KV cache positions per layer and per attention head; then, it sequentially retains key–value vectors until a stopping condition, determined by our input-insensitive threshold metric, is met. The KV cache at those remaining positions is subsequently discarded.

In line with most prior KV compression works (Li et al., 2025), our method is applied only once after input prefilling, with the primary goal of reducing memory consumption during inference and bringing improved throughput (Section 4.3).

Initial Ranking The first stage ranks all KV cache, such that the beginning of the sequence may likely contain more critical information than its latter parts, which forms the basis for downstream sequential eviction.

At this initial stage, we exploit the properties observed by prior works. First, positions at the beginning of the input generally play a more critical role in subsequent generation, known as *attention sinks* (Xiao et al., 2024; Sun et al., 2026). Second, latest positions usually receive a greater attention ratio (Gu et al., 2025). Building on the *positional bias* reported in previous works, we rank KV cache by token positions as follows.

Denote a LLM input sequence with n tokens as $X = \{x_1, x_2, \ldots, x_n\}$, where each Transformers layer originally consists of n positions of KV vectors per attention head. The initial ranking takes the first m positions and reversely takes the remaining $n - m$ positions, denoted by $\widetilde{X} = \{x_1, x_2, \ldots, x_m, x_n, x_{n-1}, \ldots, x_{m+1}\}$. m is a chosen hyperparameter that works well regardless of specific input sequences.

Despite its simplicity, the position-based ranking is shown not only effective but also particularly advantageous in terms of computational overhead, comparing to other ranking strategies we conducted in Section 4.4, therefore constituting the first stage of ReFreeKV. However, relying solely by positions does not fulfill our objective, as our experiments in Appendix E show that prior such methods such as StreamingLLM (Xiao et al., 2024) exhibit inconsistent performance across inputs, which highlights the need for a more robust KV eviction strategy.

Eviction by *Uni-Metric* With the initial ranking on KV cache, ReFreeKV then sequentially retains KV vectors, and halts upon the stopping condition by an input-insensitive threshold metric, termed *Uni-Metric*, after which the remaining cache is effectively evicted.

The design of *Uni-Metric* is then at the core of this process, which requires to signal the degradation level after removing KV cache of certain positions. we propose a metric that empirically correlates well with the performance change when discarding a position, which could **serve as a bridge to ensure a minimal degradation upon the full cache performance**. Inspired by Devoto et al. (2024), we utilize the Frobenius norm (L2 norm) of the attention matrix $A \in \mathbb{R}^{n \times n}$ as the *Uni-Metric*, denoted as $\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |A_{i,j}|^2}$. For each position i in the initially ranked sequence, we compare the Frobenius norm of the original attention matrix, $\|A\|_F$, with that of a curated attention matrix, $\|\widetilde{A}_i\|_F$, in which scores to all positions $> i$ in the ranked sequence are masked out, replicating the effect of discarding all KV cache beyond position i. Once the norm difference reaches a threshold T at position i_{prune}, the entire process terminates, retaining only the KV cache up to i_{prune} and discarding the remainder, denoted as:

$$i_{\text{prune}} = \text{argmin}_{j=1}^{n}\left(1 - \frac{\|\widetilde{A}_j\|_F}{\|A\|_F} < T\right) \qquad (1)$$

The Universal Threshold To fulfill our objective, the threshold T should ensure near lossless pruning invariant to inputs. Upon empirical search, we identify $T = 1\%$ could serve well for this purpose. Figure 2 illustrates how performance varies with changes in the norm across different domains. Preliminary studies indicate that when $T < 1\%$, performance remains comparable to the full-cache version robustly. We select $T = 1\%$ to balance minimal degradation with maximal cache eviction. The efficacy of *Uni-Metric* and its universal threshold is validated at full scale in the main experiments presented in Section 4.2 and Figure 3.

Reducing Overhead As the input sequence length n increases, the time and space overhead for norm calculation on the attention matrix grows by $O(n^2)$. To reduce the computational scale, we seek to use an approximate norm calculation by $O(n)$. Instead of using the full attention A, we reduce A by taking the average of its last k rows to a single attention vector $A' \in \mathbb{R}^{1 \times n}$. The score for a

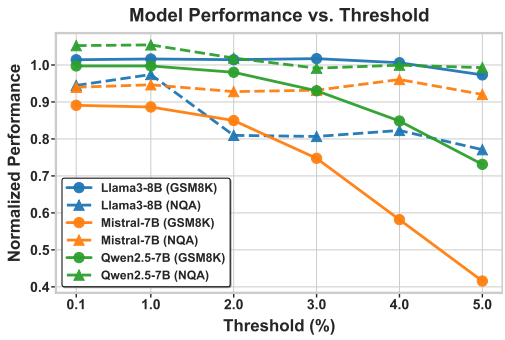

Figure 2: Performance trends of Llama3-8B, Mistral-7B, and Qwen2.5-7B across varying *Uni-Metric* thresholds. The x-axis represents the threshold percentage (0.1% to 5%), and the y-axis denotes the performance score normalized by the full-cache performance.

position $i \in [1, n]$ in A' is denoted as:

$$A'[i] = \frac{\sum_{j=k}^{n} A_{i,j}}{\sum_{j=k}^{n} \mathbf{1}_{\{A_{i,j} \neq 0\}}} \tag{2}$$

The Two-Stage Logistics The rationale behind ReFreeKV 's two-stage process is that the precise determination of optimal pruning positions in a sequence is a *combinatorial* problem. By adopting a approximated ranking stage along with a linear search, the computation becomes tractable. The entire procedure is further optimized through parallel operations, as described below.

3.3 Implementation Details

As ReFreeKV is conceptually a sequential search process, the design of ReFreeKV allows efficient implementation by PyTorch's operators, such that the stopping positions of all layers are directly identified in parallel without explicit looping operations. Latency and throughput analyses in Section 4.3 demonstrate that ReFreeKV matches prior popular KV pruning approaches, with negligible latency overhead and improved throughput compared to the baselines.

Specifically, the pruning position i_{prune} can be determined directly by the combination of Torch `cumulative-sum` and `where` operators. Given the reduced attention matrix after the initial ranking, A'_{rank}, we compute the cumulative square-sum of each element, such that $A'_{cumsum}[i]$ represents the Frobenius Norm of the attention matrix after removing all cache to the right of position i in A'_{rank}:

$$\|\widetilde{A'_i}\|_F = A'_{cumsum}[i] = \sqrt{\sum_{k=1}^{i} \left(A'_{\text{rank}}[k]\right)^2} \tag{3}$$

We then divide $\|\widetilde{A'_i}\|_F$ by the full norm of A'_{rank} to determine the norm difference as in Eq (1). The torch `where` operation allows us to directly identify the leftmost position that satisfies the 1% universal threshold, ultimately yielding the set of positions for which the KV cache is called to retain. The overall pruning process of ReFreeKV is further presented in Algorithm 1.

Retaining Bottom Layers In our experiments, we identify that the LLM's first two Transformers layers have a relatively uniform attention distribution, usually requiring to retain most of the cache positions, which aligns with prior studies that early layers are important for basic semantic understanding (Reif et al., 2019; Skean et al., 2025). For simplicity and robustness, we always retain full KV cache of the first two layers in our implementation. We provide more studies on retaining bottom layers in Appendix B.

Running at Scale Supporting batch sizes > 1, ReFreeKV is able to perform the entire pruning process for each sample in parallel. To achieve this, we pad the shorter cache segments and update the attention masks accordingly, allowing the LLM to ignore the padded KV positions. The padding operation has a negligible impact on overall performance. Meanwhile, in popular LLM inference engines, e.g. vLLM (Kwon et al., 2023), it is possible to allocate separate KV cache size for each sample, which aligns well with ReFreeKV.

4 Experiments

4.1 Experimental Settings

LLM Backbones Our experiments are conducted with three LLM families of different model sizes: Llama3-Instruct with size of 8B/70B, Mistral-7B-Instruct-V0.3, and Qwen2.5-Instruct with size of 7B/32B/72B. We implement our ReFreeKV upon the released codebase of SnapKV*. For the reduced attention matrix A', we set $k = 1$ in practice (ablation provided in Section 4.4), and set $m = 4$ for the initial ranking stage.

Datasets For comprehensive evaluation, we evaluate ReFreeKV on datasets of both short and long context length of different domains, including mathematics, science, and commonsense reasoning on GSM8K (Cobbe et al., 2021), GPQA (Rein et al.,

* https://github.com/FasterDecoding/SnapKV

	Methods	Math & Science			CR		Single-Doc QA		Multi-Doc QA		Summarization		FSL	Code	Avg.
		GSM8K	GPQA	TheoQA	ThQA	CoQA	NrtvQA	Qasper	2WikiMQA	Musique	QMSum	M-News	TriviaQA	Lcc	
Llama3-8B-Instruct	Full	75.28	29.02	21.29	25.59	52.74	24.06	43.91	35.33	14.77	22.27	27.37	70.26	19.16	100%
	Ours$_{k=1}$	**76.50**	**30.13**	23.03	**26.09**	**52.86**	23.44	37.38	**36.21**	**15.96**	21.95	**27.88**	64.24	19.31	**+0.12%**
	Budget	*93.2%*	*86.7%*	*92.8%*	*78.7%*	*81.5%*	*48.7%*	*46.4%*	*45.8%*	*43.7%*	*15.0%*	*76.4%*	*41.0%*	*78.0%*	*63.68%*
	H2O$_{0.9}$	74.60	28.13	21.69	25.41	52.83	**24.55**	41.73	33.95	15.23	**22.51**	27.64	69.77	19.13	-0.39%
	SLM$_{0.9}$	72.78	28.79	20.75	25.15	52.83	23.75	43.63	32.68	15.86	22.48	27.21	69.97	**19.66**	-0.59%
	SnapKV$_{0.9}$	70.20	27.90	20.08	25.38	52.72	24.04	**43.93**	33.92	15.56	22.49	27.65	**70.58**	19.05	-1.07%
	PyramidKV$_{0.9}$	75.44	28.79	**24.46**	24.72	52.16	23.64	43.39	32.10	14.11	22.22	23.70	70.48	19.05	-1.59%
	CAKE$_{0.9}$	74.23	27.23	23.16	22.87	51.18	21.32	43.78	34.65	14.20	21.19	22.33	69.22	19.22	-4.17%
	H2O$_{0.5}$	31.08	2.90	16.47	17.99	52.37	23.14	43.05	32.79	15.77	22.79	26.24	69.83	19.18	-16.17%
	SLM$_{0.5}$	3.41	6.03	8.84	18.75	44.68	20.94	37.73	31.20	15.29	21.80	25.93	67.79	19.12	-24.73%
	SnapKV$_{0.5}$	12.96	7.81	16.87	19.52	52.84	23.40	43.93	33.98	15.94	22.67	26.07	69.66	19.19	-15.57%
	H2O$_{0.2}$	2.43	2.46	6.56	20.67	50.91	23.54	42.00	32.50	15.67	22.16	23.68	69.88	19.13	-23.33%
	SLM$_{0.2}$	3.21	3.35	8.82	18.02	40.11	20.12	36.45	29.91	15.01	21.11	24.68	66.17	18.69	-28.21%
	SnapKV$_{0.2}$	1.29	1.12	5.89	20.36	52.00	23.10	42.78	32.98	14.01	22.45	24.14	69.63	19.34	-24.45%
	PyramidKV$_{0.2}$	1.36	1.79	4.82	20.24	51.71	24.31	41.90	33.81	12.56	21.11	27.01	68.43	18.19	-25.33%
	CAKE$_{0.2}$	1.67	4.46	7.93	21.17	50.66	22.01	42.18	33.02	14.93	20.15	24.48	69.92	19.10	-23.47%
Mistral-7B-Instruct	Full	33.36	29.24	6.83	20.82	39.68	28.74	37.80	33.87	22.88	22.19	22.94	86.87	16.08	100%
	Ours$_{k=1}$	**31.54**	**29.02**	6.71	**20.81**	**39.80**	27.20	38.93	33.46	22.15	**22.39**	22.67	**86.81**	15.33	**-1.50%**
	Budget	*89.6%*	*90.5%*	*84.2%*	*92.3%*	*84.1%*	*78.0%*	*97.9%*	*86.7%*	*74.4%*	*84.3%*	*89.1%*	*87.2%*	*89.4%*	*86.75%*
	H2O$_{0.9}$	19.18	24.11	6.96	20.61	39.74	27.36	38.38	35.23	23.24	22.13	23.41	86.46	16.01	-4.29%
	SLM$_{0.9}$	31.16	27.68	5.89	19.88	39.15	27.26	37.66	35.06	21.94	22.31	21.73	86.63	13.67	-4.44%
	SnapKV$_{0.9}$	27.60	25.67	**7.10**	20.23	39.76	27.43	38.27	35.66	22.99	22.16	23.31	86.46	16.07	-1.90%
	PyramidKV$_{0.9}$	22.74	26.75	5.35	20.34	38.43	**27.60**	38.27	**37.62**	**25.28**	22.18	**24.64**	85.53	**16.37**	-3.15%
	CAKE$_{0.9}$	25.02	24.10	6.12	20.21	38.82	27.49	**39.19**	32.96	23.55	22.22	24.51	86.36	15.80	-4.14%
	H2O$_{0.5}$	2.50	8.26	5.89	20.28	38.53	27.62	36.97	34.31	23.17	22.32	22.60	86.52	16.45	-14.31%
	SLM$_{0.5}$	2.43	6.47	1.14	18.27	32.76	24.91	33.80	32.73	20.68	21.48	18.54	86.45	13.58	-27.61%
	SnapKV$_{0.5}$	3.03	8.93	6.83	19.03	39.22	27.15	38.27	34.60	22.94	21.93	22.68	86.31	16.18	-13.41%
	H2O$_{0.2}$	1.52	4.46	4.02	18.45	35.55	27.28	33.45	34.36	23.21	21.80	20.93	85.99	14.82	-21.25%
	SLM$_{0.2}$	1.21	1.79	0.54	18.35	25.45	24.89	28.39	29.53	16.74	20.60	16.75	80.12	14.27	-35.48%
	SnapKV$_{0.2}$	1.44	0.20	3.35	17.81	37.44	26.76	36.02	33.90	23.05	22.13	20.88	84.27	16.20	-22.18%
	PyramidKV$_{0.2}$	1.14	0.58	1.40	19.24	31.20	26.19	34.53	37.25	24.55	22.24	22.18	86.58	15.76	-23.75%
	CAKE$_{0.2}$	4.55	1.03	2.09	18.21	30.07	26.25	36.03	32.39	22.79	21.53	23.96	86.61	15.00	-24.05%
Qwen2.5-7B-Instruct	Full	88.02	31.70	29.85	24.55	61.43	*20.81*	43.17	47.15	30.70	23.64	24.24	87.64	2.44	100%
	Ours$_{k=1}$	88.02	**31.25**	**30.39**	**24.48**	60.01	20.66	42.74	47.20	29.56	22.64	24.04	**87.65**	**3.58**	**+2.63%**
	Budget	*90.7%*	*88.8%*	*86.5%*	*69.2%*	*84.1%*	*65.1%*	*69.2%*	*73.3%*	*64.4%*	*70.6%*	*85.4%*	*56.6%*	*84.4%*	*76.02%*
	H2O$_{0.9}$	83.47	26.56	30.25	24.40	**61.29**	20.85	43.26	**47.90**	**30.73**	23.42	**24.35**	87.57	2.45	-1.46%
	SLM$_{0.9}$	**88.93**	30.80	28.25	24.30	60.91	19.86	42.54	46.02	28.75	23.06	23.42	87.03	3.05	-0.41%
	SnapKV$_{0.9}$	80.14	30.13	28.11	24.30	61.26	20.90	**43.31**	47.81	30.69	23.90	24.29	87.57	2.55	-1.01%
	PyramidKV$_{0.9}$	84.15	27.68	26.78	23.51	50.34	27.60	42.27	37.62	25.28	22.18	21.65	87.53	3.37	-2.76%
	CAKE$_{0.9}$	85.67	30.23	29.28	23.52	58.27	**29.27**	33.73	46.60	30.32	**23.91**	23.89	90.14	2.41	-0.07%
	H2O$_{0.5}$	34.12	14.73	20.88	23.10	59.69	21.59	43.37	47.60	30.81	21.00	22.95	85.54	2.24	-13.47%
	SLM$_{0.5}$	3.26	1.34	10.58	23.75	49.87	19.29	36.45	42.42	25.33	21.24	22.77	87.33	3.40	-23.56%
	SnapKV$_{0.5}$	20.77	12.28	22.76	22.25	60.21	20.93	43.22	47.45	30.61	23.76	22.70	87.57	2.21	-14.39%
	H2O$_{0.2}$	5.91	3.57	17.40	21.70	55.39	21.41	40.30	45.84	29.72	23.04	20.84	87.99	2.15	-21.77%
	SLM$_{0.2}$	1.06	4.24	4.82	23.85	40.92	18.17	29.22	39.27	21.71	20.32	18.94	77.80	1.46	-37.22%
	SnapKV$_{0.2}$	2.58	7.37	15.13	21.52	57.06	21.40	40.93	46.96	29.87	23.58	20.69	87.10	2.44	-20.27%
	PyramidKV$_{0.2}$	9.70	8.57	9.60	26.42	51.19	26.42	39.48	39.54	25.73	21.02	21.65	85.31	1.76	-23.47%
	CAKE$_{0.2}$	7.58	9.11	5.56	23.66	49.19	30.20	43.04	46.35	29.00	23.33	22.94	85.45	2.89	-16.98%

Table 2: Performance of ReFreeKV and its comparison with five KV pruning methods on 13 datasets. **Bold** numbers indicate the best results **aside from full-cache**. *Italics* represent the real budget utilized by ReFreeKV. The correspondence between abbreviations and their full names of datasets can be found in Appendix C. **Avg.** calculates the mean ratio of the model's performance using different KV cache compression methods to its performance with the full cache. All average results (except for budget) are adjusted by subtracting 1 to provide a more intuitive understanding of the effectiveness of different methods.

2023), TheoremQA (Chen et al., 2023), TruthfulQA (Lin et al., 2022), and CoQA (Reddy et al., 2019). We also include tasks from Longbench (Bai et al., 2024) with 8 datasets spanning document comprehension, summarization and coding. Appendix C provides a detailed description and statistics for all **13 datasets**, along with how they are utilized in our experiments.

Evaluation Protocol Our evaluation primarily assesses ReFreeKV 's ability to preserve full-cache performance with its automatic pruning budgets. Accordingly, we compare ReFreeKV against its full-cache scores, and also report the average compression ratio for each dataset.

Additionally, we compare with five prior KV cache pruning methods with varied budget sizes, including: Heavy Hitter Oracle (**H2O**) (Zhang et al., 2023), StreamingLLM (**SLM**) (Xiao et al., 2024), **SnapKV** (Li et al., 2024), **PyramidKV** (Cai et al., 2024) and **CAKE** (Qin et al., 2025). By evaluating performance consistency under fixed KV cache budgets of 90%, 50%, and 20%, we further illustrate the limitations arising from the input-specific

threshold dependence.

Lastly, we also include a concurrent work **Twilight** (Lin et al., 2025), which does not rely on a budget threshold but employs a top-p-inspired metric for adaptive token selection. It is worth noting that the p value remains a hyperparameter to be tuned across models and inputs.

4.2 Main Results

The main experimental results are shown in Table 2. For comparison with Twilight, we separately report the results in Table 3 due to its different hyperparameter type. Based on Table 2, we can draw the following observations.

• ReFreeKV is able to fulfill our objective, capable of performing near-lossless dynamic compression across different models, varying input lengths, and diverse task types. Interestingly, with Llama3-8B and Qwen2.5-7B, ReFreeKV even surpasses the full-cache performance by 0.12% and 2.63% respectively, utilizing an average of 63.68% and 76.02% KV cache. With Mistral, ReFreeKV also manages to achieve near 15% compression with a relatively small 1.5% performance reduction. These results indicate that our proposed method can strike for real-world scenarios with no bother by input-specific budget thresholds.

• In stark contrast, previous methods achieve a consistent full-cache performance only when manually determined a high budget ratio, e.g. 90%. However, when the budget is reduced, e.g. 50% and 20%, the degradation can become severe on certain datasets, distinct from ReFreeKV that automatically adjusts the pruning to always prioritize full-cache performance. Theoretically, one could tune the budget for each dataset that achieves minimal degradation, but this is generally infeasible for real-world open-domain instructions.

• ReFreeKV naturally reflects the *difficulty* of the generation task. As in Table 2, the dynamic budget ratio is high on Math&Science datasets (over 90%), while much lower on QA or Summarization datasets (as low as 15%). This observation is in line with our intuition, where inference on concise but hard tasks, such as math problems, requires more context and more precise calculation, resulting in higher budget allocation. From this aspect, our method design serves beyond for memory efficiency, but could be potentially leveraged for input analysis in a broader scope.

• Besides the dynamic compression, ReFreeKV also outperforms the three 90%-budget baselines, while itself uses less than 90% budget. Though, It is worth reiterating that the goal of this work is not to propose yet another KV cache pruning method that targets the best possible compression ratio under specific conditions. Instead, we seek to lift the threshold constraints and advocate for robust KV pruning that generalizes to arbitrary inputs.

	GSM8K	NQA	Qasper	2WQA	Musique
Llama3-8B	75.28	24.06	43.91	35.33	14.77
+ Ours$_{k=1}$	**76.50**	**23.44**	37.38	**36.21**	**15.96**
+ Twi$_{p=0.95}$	76.04	23.37	43.08	36.18	14.92
Mistral-7B	33.36	28.74	37.80	33.87	22.88
+ Ours$_{k=1}$	**31.54**	**27.20**	**38.93**	**33.46**	22.15
+ Twi$_{p=0.85}$	30.33	27.17	38.90	33.23	22.92

Table 3: Performance comparison between ReFreeKV and Twilight across five datasets using Llama3-8B-Instruct and Mistral-7B-V0.3. **Bold** numbers indicate the best results aside from full-cache.

As separately reported in Table 3, we compare ReFreeKV with Twilight's reported performance across the GSM8K, NarrativeQA, 2WikiMQA, and Musique datasets. Regarding the hyperparameter for Twilight, we adopt their reported optimal p value: $p = 0.95$ for Llama3-8B and $p = 0.85$ for Mistral. As shown in the results, ReFreeKV achieves performance comparable to Twilight on both models. Notably, both methods frequently surpass the full-cache baseline across these datasets. Though, as Twilight requires different optimal p values for each model, hyperparameter tuning and selection still remain necessary. Nonetheless, both ReFreeKV and Twilight share the same principle of adaptive pruning. We hope our proposed objective and method could further advance the research in this direction.

4.3 Efficiency Analysis

End-to-End Latency In this section, we conduct a quantitative analysis of the latency of ReFreeKV and its overall impact on inference time, beyond the memory savings from KV-cache pruning. We compare runtime on six datasets using Llama3-8B and Llama3-70B, evaluating ReFreeKV against three baselines under a 50% budget setting. Table 4 reports both the latency of the pruning operation (**Prune**) and the average generation time per sample after the prefilling stage (**Overall**).

The results show that the pruning latency of ReFreeKV is comparable to prior methods. Notably, by automatically adapting to achieve higher compression ratios, ReFreeKV attains the best overall

	GSM8K		CoQA		NarrativeQA		Musique		QMSum		TriviaQA		Avg.	
	Overall	Prune	Overall	Prune	Overall	Prune	Overall	Prune	Overall	Prune	Overall	Prune	Overall	Prune
Llama3-8B-Instruct														
Full	4.693	—	**0.289**	—	3.441	—	3.659	—	5.458	—	2.717	—	3.376	—
Ours$_{k=1}$	**4.638**	0.034	0.331	0.033	**3.159**	**0.255**	**3.658**	0.264	**5.330**	0.241	2.684	0.211	**3.300**	0.173
H2O$_{0.5}$	4.686	0.020	0.290	**0.018**	3.243	0.264	3.679	0.266	5.398	0.249	**2.612**	0.237	3.318	0.176
SLM$_{0.5}$	6.071	**0.006**	0.301	0.030	3.230	0.257	3.784	**0.259**	5.454	**0.234**	2.628	**0.208**	3.578	**0.166**
SnapKV$_{0.5}$	5.874	0.021	0.285	0.019	3.307	0.265	3.721	0.266	5.519	0.264	2.650	0.265	3.559	0.183
Llama3-70B-Instruct														
Full	17.504	—	1.167	—	6.285	—	**6.565**	—	13.993	—	5.302	—	8.469	—
Ours$_{k=1}$	**15.975**	0.154	1.253	0.290	**6.042**	2.345	7.412	2.352	**13.835**	3.028	**5.224**	1.602	**8.290**	1.623
H2O$_{0.5}$	19.062	**0.114**	**1.059**	0.240	6.788	2.298	7.082	2.307	14.360	3.712	5.566	1.308	8.986	1.663
SLM$_{0.5}$	22.079	0.150	1.138	**0.229**	6.804	**2.069**	7.107	**2.053**	14.734	3.700	5.576	1.367	9.573	1.595
SnapKV$_{0.5}$	16.517	0.117	1.116	0.241	6.795	2.474	7.085	2.468	13.846	**2.793**	5.562	**1.083**	8.487	**1.529**

Table 4: The average inference time and pruning time of Llama3 with size of 8B/70B across six datasets, measured in seconds, with lower values indicating better performance.

generation time in 8 out of 12 comparisons, indicating a clear advantage in end-to-end generation speed. This trend remains consistent across model scales, further underscoring the efficiency aspect of ReFreeKV.

Batched Processing We further conduct a comprehensive analysis of latency and throughput across varying batch sizes, as reported in Table 11. Following the FastGen setup, we use inputs from NarrativeQA, and perform end-to-end latency evaluations on a single A100 GPU, with the standard HuggingFace (HF) Accelerate library as the baseline. ReFreeKV consistently improves throughput over naive generation by 10–20%. Especially, it retains its performance edges robustly with increasing batch sizes.

Overall, Table 4 and Table 11 highlight the efficiency of ReFreeKV. The additional latency introduced by pruning is minimal, owing to the trivial overhead of the lightweight two-stage design. Meanwhile, the reduced KV cache lowers attention computation costs, resulting in improved inference latency and overall throughput in the end.

4.4 Ablation Studies

We conduct ablation studies to investigate the impact of various configurations of ReFreeKV. We use Llama-3-8B-Instruct and perform experiments across five datasets, with results presented in Table 5. Appendix D provides additional results and analysis from our ablation studies.

Attention Matrix Reduction The reduced attention matrix A' in Section 3.2 aggregates attention scores from the last k rows. The upper part of Table 5 illustrates the model's performance and the actual budget when setting $k = 1$, $1\%n$, $5\%n$, and $10\%n$ (n being the number of input tokens). It is clear that setting k as 1 achieves a significantly reduced budget, thus a higher compression ratio, with almost no change in performance compared to $1\%n$ and $5\%n$. On the other hand, while $10\%n$ can compress more KV cache, it fails to maintain performance (for instance, on NarrativeQA, the former achieves a performance of 9.74 using 32% of the budget, whereas the latter scores 21.44 using 48.7% of the budget). What's even better is that since $k = 1$ requires the least amount of computation, relying solely on the scores from the last row, the complexity of obtaining A' becomes $O(1)$, independent of the sequence length. The advantages of both high efficacy and low overhead make $k = 1$ a solid design choice for calculating the norm metric.

	GSM8K	CoQA	NQA	Musique	QMSum
Full	75.28	52.74	24.06	14.77	22.27
Pos$_{t=1\%}$	*Performance Using Different k*				
$k = 1$	76.50	52.86	23.44	15.96	21.95
Budget	*93.2%*	*81.5%*	*48.7%*	*43.7%*	*15.0%*
$k = 1\%n$	76.19	52.87	23.17	15.40	21.94
Budget	*95.1%*	*94.8%*	*77.3%*	*77.2%*	*59.5%*
$k = 5\%n$	75.59	52.75	21.62	13.71	21.43
Budget	*98.6%*	*96.2%*	*45.1%*	*30.0%*	*33.1%*
$k = 10\%n$	76.72	52.85	9.74	13.67	21.11
Budget	*96.3%*	*96.8%*	*32.0%*	*19.7%*	29.3%
$k = 1$	*Performance with Different Ranking Method*				
Attn$_{t=1\%}$	2.35	43.68	13.37	9.58	17.62
Budget	*20.0%*	*9.7%*	*6.4%*	*6.4%*	*6.4%*
Attn$_{t=0.01\%}$	54.66	51.58	17.45	14.79	20.90
Budget	*61.7%*	*28.6%*	*6.7%*	*7.9%*	*7.1%*

Table 5: Performance comparison for ablation studies in Section 4.4: different k values for attention matrix reduction, and using an alternative initial ranking strategy by attention scores with different thresholds.

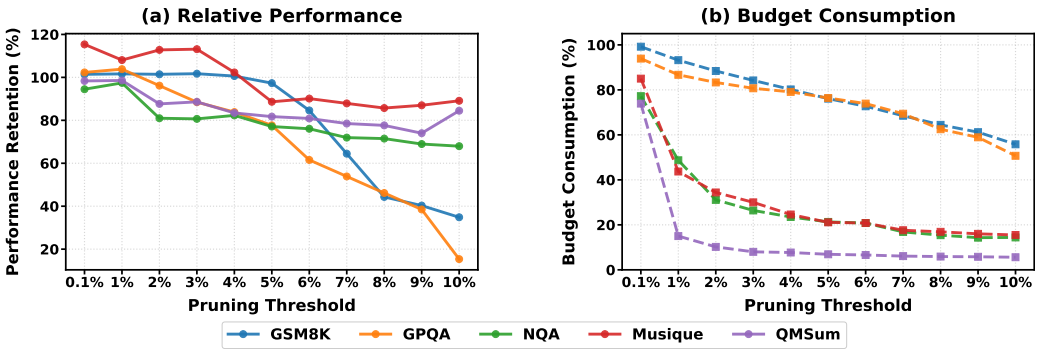

Figure 3: **Performance vs. Efficiency Trade-off.** (a) Performance retention across five datasets (solid lines). (b) Computational budget consumption (dashed lines) relative to the dense baseline. The shared legend indicates the datasets. Results show that setting the universal threshold to 1% could well balance between performance and memory budget, as it maintains robust full-cache performance while substantially reducing KV cache.

Initial Ranking Strategies Apart from the position-based ranking described in Section 3.2, we also investigate other alternatives, such as ranking by each token's average attention score received from other tokens, similar to the approach in H2O (Zhang et al., 2023). The results, reported at the bottom of Table 5, show that across different attention-score thresholds, this attention-based ranking fails to maintain robust final performance, suggesting that raw attention scores alone are not reliable indicators of KV-cache importance. Empirical validation supports that position ranking is an appealing choice, offering both superior efficacy and efficiency.

The Universal Threshold With the universal threshold for the attention norm difference set to 1%, we further examine the effects of both smaller and larger values, as shown in Figure 3. Intuitively, a larger threshold yields higher compression at the cost of potential performance degradation, while a smaller threshold has the opposite effect.

When the threshold is reduced to 0.1%, the average budget increases as expected, yet model performance shows negligible improvement. In contrast, increasing the threshold to 10% results in more aggressive KV-cache pruning but leads to substantial performance degradation. These observations suggest that 1% provides a reasonable and robust trade-off for general use across models and tasks.

Generalization We further evaluate whether our design and hyperparameters generalize to LLMs of different scales. As shown in Table 6, ReFreeKV applied to Llama3-70B and Qwen2.5-32B/72B consistently achieves near full-cache performance under the same configuration used in Table 2. Notably, the average compression ratio improves on datasets with longer contexts, reaching close to 50%, while maintaining comparable performance across model sizes and datasets without any threshold tuning.

	GSM8K	CoQA	NQA	Musique	QMSum
Llama3-70B-Instruct					
Full	89.69	60.36	27.15	29.31	22.52
Ours	89.76	60.38	26.94	28.88	22.30
Budget	*93.3%*	*77.2%*	*58.7%*	*60.8%*	*52.9%*
Qwen2.5-32B-Instruct					
Full	91.36	58.03	24.78	40.04	22.84
Ours	91.81	57.29	22.65	40.54	22.52
Budget	*91.6%*	*85.0%*	*68.2%*	*74.4%*	*76.8%*
Qwen2.5-72B-Instruct					
Full	90.22	54.14	24.36	42.13	23.93
Ours	90.30	54.19	24.10	41.70	23.31
Budget	*90.9%*	*87.6%*	*63.4%*	*65.0%*	*68.6%*

Table 6: Performance of different LLMs with scales of 70B, 32B, and 72B across five datasets with the exact same ReFreeKV configuration as in Table 2, demonstrating the generalization of ReFreeKV.

5 Conclusion

In this study, we introduce a new KV cache compression objective that lifts the threshold dependency, so to achieve input-insensitive pruning for robust inference performance. Towards this objective, we propose a novel method, termed ReFreeKV, which employs a straightforward yet effective two-stage KV cache pruning process. Comprehensive experiments conducted across diverse datasets, encompassing a variety of tasks, context lengths and LLM models, demonstrate that ReFreeKV achieves near-lossless compression robustly without involving any input-specific thresholds, while able to deliver notable KV cache reduction.

Limitations

The primary limitation of ReFreeKV lies in the gap between its achieved compression ratio and the true optimal budget. As shown in Table 2, in certain scenarios (e.g., QMSum with Mistral-7B), ReFreeKV retains an 84.3% budget, whereas a 50% budget remains viable without performance degradation. We view this gap as an opportunity for future work to enable more aggressive KV-cache compression while still satisfying the full-cache objective under dynamic budgets.

Another limitation is that, although ReFreeKV demonstrates near-lossless compression empirically, it does not provide formal guarantees on performance degradation. As reported in Table 2, while ReFreeKV even surpasses full-cache performance for Llama3-8B and Qwen2.5-7B, it incurs a small degradation (1.5%) on Mistral-7B. Developing more principled approaches to further improve robustness remains an important direction for future research.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *CoRR*, abs/2307.03109.

Jinhan Chen, Jianchun Liu, Hongli Xu, Xianjun Gao, and Shilong Wang. 2025. Sablock: Semantic-aware KV cache eviction with adaptive compression block size. *CoRR*, abs/2510.22556.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7889–7901. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. 2024. A simple and effective l_2 norm-based strategy for KV cache compression. *CoRR*, abs/2406.11430.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1074–1084. Association for Computational Linguistics.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. 2024. Ada-kv: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *CoRR*, abs/2407.11550.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive KV cache compression for llms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min

Lin. 2025. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian J. McAuley. 2023. Longcoder: A long-range pretrained language model for code completion. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 12098–12107. PMLR.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Gregory Kamradt. 2023. Needle In A Haystack - pressure testing LLMs. *Github*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328.

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

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole HU, Wei Dong, Li Qing, and Lei Chen. 2025. A survey on large language model acceleration based on KV cache management. *Transactions on Machine Learning Research*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. 2025. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. *CoRR*, abs/2502.02770.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3214–3252. Association for Computational Linguistics.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Shervin Minaee, Tomás Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *CoRR*, abs/2402.06196.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. 2025. CAKE: cascading and adaptive KV cache eviction with layer preferences. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge. *Trans. Assoc. Comput. Linguistics*, 7:249–266.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*.

Shangwen Sun, Alfredo Canziani, Yann LeCun, and Jiachen Zhu. 2026. The spike, the sparse and the sink: Anatomy of massive activations and attention sinks. *Preprint*, arXiv:2603.05498.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, and Mi Zhang. 2024. D2O: dynamic discriminative operations for efficient generative inference of large language models. *CoRR*, abs/2406.13035.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. 2024. Model tells you where to merge: Adaptive KV cache merging for llms on long-context tasks. *CoRR*, abs/2407.08454.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hui Zeng, Daming Zhao, Pengfei Yang, WenXuan Hou, Tianyang Zheng, Hui Li, Weiye Ji, and Jidong Zhai. 2026. Lethe: Layer- and time-adaptive KV cache pruning for reasoning-intensive LLM serving. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 28103–28112. AAAI Press.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics.

A Full Algorithm of ReFreeKV

Algorithm 1 ReFreeKV

Input: Prompt, Threshold t
Output: Compressed KV Cache
Create Empty List K_c, V_c
for *Transformer Layer* L_i *in LLM* **do**
  $Q^i, K^i, V^i \leftarrow L_i(\text{Prompt})$
  $R^i \leftarrow$ Postion-Based Importance Rank
  $A^i_{last} \leftarrow \text{Attention}(Q^i[\ldots, -1, :], K^{iT})$
  $F^i_b \leftarrow \text{Frobenius}(A^i_{last})$
  $A^i_{last} \leftarrow \text{Square}(A^i_{last})$
  Reorder A^i_{last} by Rank R
  $A^i_{cumsum} \leftarrow \text{Cumsum}(A^i_{last})$
  $A^i_{cumsum} \leftarrow \text{Sqrt}(A^i_{cumsum})$
  $A^i_{ratio} \leftarrow (F^i_b - A^i_{cumsum})/F^i_b$
  $\text{Index}^i \leftarrow \text{Max}(\text{Where}(A^i_{ratio} <= t))$
  $K^i_c \leftarrow \text{Compress } K^i \text{by} R^i[I:]$
  $V^i_c \leftarrow \text{Compress } V^i \text{by} R^i[I:]$
  Append K^i_c, V^i_c to K_c, V_c
end for
return K_c, V_c

B Retain Bottom LLM Layers

As described in Section 3.2, the KV-cache pruning in ReFreeKV begins from the third LLM layer; here, we empirically justify this design choice. We apply ReFreeKV with Llama3-8B and evaluate it on GSM8K and CoQA. We first present a case study, followed by a comparison examining the impact of not retaining/freezing the full KV cache in the first two layers.

As shown in Table 7, when the threshold is set to 1% and no layers are frozen, the outputs for two examples on GSM8K and CoQA are both incorrect and lack logical coherence, despite the overall budget exceeding 90% in each case. Examining the per-layer budgets reveals that layers 1 and 2 have relatively low budgets, whereas layers 3 through 31 exhibit uniformly high budgets, with the final layer dropping again. We hypothesize that, in the first two layers, the model has not yet formed sufficiently discriminative attention patterns to identify truly important tokens, leading to a relatively uniform attention distribution. This can result in the premature eviction of important tokens, preventing later layers from accessing critical information and ultimately causing generation failures.

Based on the above case study, we further explore retaining specific layers to identify an optimal configuration. We evaluate different layer-freezing strategies, with partial results summarized in Table 8. We observe that freezing the first two layers can strike a balance between model performance and budget. In contrast, additionally retaining the 31st layer yields negligible performance gains while incurring a higher budget. Accordingly, ReFreeKV applies KV-cache compression starting from the third layer.

GSM8K	
Input	A robe takes 2 bolts of blue fiber a-nd half that much white fiber. How many bolts in total does it take?
Budget	*Layer 1*: 67.71 *Layer 2*: 82.29 *Layer 3~31:* 95.83 *Layer 32*: 37.50 **Avg.**: 93.90
Output	I have determined by answering the answer to the format of bolts bolts b-olts...(repeat)
Ground-Truth	3
CoQA	
Input	You are given a story and a question. Answer the question as concisely as you can...Question: What color was Cotton?
Budget	*Layer 1*: 42.14 *Layer 2*: 42.54 *Layer 3~31*: 99.19 *Layer 32*: 79.64 **Avg.**: 95.03
Output	Question: Question: What is the que-stion: What is the question:
Groud-Truth	White

Table 7: Case Study (Appendix B).

Datasets		**None**	**0**	**0,1**	**0,1,2**	**0,1,2,3**	**0,1,31**
GSM8K	Score	3.9	20.32	76.50	76.65	76.35	76.57
	Budget	*86.7%*	*87.0%*	*93.2%*	*94.1%*	*94.7%*	*96.6%*
CoQA	Score	20.89	51.82	52.86	51.58	53.32	52.46
	Budget	*75.5%*	*77.8%*	*81.5%*	*82.8%*	*83.0%*	*82.5%*

Table 8: Performance of Llama3-8B-Instruct on GSM8K and CoQA using ReFreeKV varying frozen layer configurations. The column headers indicate which layers are frozen/retained during inference.

C Datasets Used in Experiments

In this section, we provide a comprehensive overview of all the tasks and datasets utilized in the

	GSM8K	GPQA	CoQA	NrtvQA	QMSum	TriviaQA
Full	75.28	29.02	52.74	24.06	22.27	70.26
Ours$_{k=1}$	76.50	30.13	52.86	23.44	21.95	64.24
Budget	*93.2%*	*86.7%*	*81.5%*	*48.7%*	*15.0%*	*41.0%*
SLM$_{0.9}$	72.78	28.79	52.83	23.75	22.48	69.97
SLM$_{F0.9}$	76.42	29.24	52.89	22.95	22.47	69.10
SLM$_{0.5}$	3.41	6.03	44.68	20.94	21.80	67.79
SLM$_{F0.5}$	4.55	3.13	44.59	21.31	21.88	68.04
SLM$_{0.2}$	3.21	3.35	40.11	20.12	21.11	66.17
SLM$_{F0.2}$	1.21	1.12	38.44	19.52	20.61	65.69

Table 9: Performance comparison between ReFreeKV and StreamingLLM (**SLM**) with the first two layers frozen (**SLM**$_F$) under difference KV cache budgets (0.9, 0.5 and 0.2).

Methods	GSM8K	CoQA	NQA	Musique	QMSum
Full	75.28	52.74	24.06	14.77	22.27
Pos$_{t=1\%}$	*Performance Using Different k*				
k = 1	76.50	52.86	23.44	15.96	21.95
Budget	*93.2%*	*81.5%*	*48.7%*	*43.7%*	*15.0%*
k = 1%n	76.19	52.87	23.17	15.40	21.94
Budget	*95.1%*	*94.8%*	*77.3%*	*77.2%*	*59.5%*
k = 2%n	76.19	52.82	22.87	14.34	21.74
Budget	*96.0%*	*96.1%*	*69.8%*	*62.0%*	*47.1%*
k = 3%n	75.82	52.80	23.03	13.61	21.56
Budget	*98.6%*	*95.7%*	*60.3%*	*46.0%*	*39.5%*
k = 4%n	75.21	52.76	22.96	13.45	21.47
Budget	*98.8%*	*95.8%*	*51.9%*	*35.2%*	*35.4%*
k = 5%n	75.59	52.75	21.62	13.71	21.43
Budget	*98.6%*	*96.2%*	*45.1%*	*30.0%*	*33.1%*
k = 6%n	75.66	52.81	21.95	14.33	20.94
Budget	*98.5%*	*96.6%*	*40.4%*	*25.8%*	*31.6%*
k = 7%n	76.57	52.88	21.18	13.66	20.94
Budget	*98.0%*	*96.7%*	*37.3%*	*22.5%*	*30.6%*
k = 8%n	76.35	52.86	20.36	13.75	21.11
Budget	*97.4%*	*96.7%*	*35.1%*	*20.9%*	*30.1%*
k = 9%n	76.65	52.84	19.99	13.66	20.76
Budget	*96.8%*	*96.8%*	*33.4%*	*19.9%*	*29.5%*
k = 10%n	76.72	52.85	9.74	13.67	21.11
Budget	*96.3%*	*96.8%*	*32.0%*	*19.7%*	29.3%

Table 10: Performance comparison with different k for reducing the full attention matrix, expanded from Table 5.

experiments of this paper.

Math & Science This task evaluates the model's ability to tackle mathematical and scientific problems. By directly inputting questions and comparing the model's output with the correct answers, we calculate the model's *Accuracy* on these datasets: **GSM8K** is a dataset for evaluating model's math-solving skills, featuring 8,000 elementary-level math word problems requiring basic arithmetic and reasoning. **GPQA** tests model's understanding of physics concepts and problem-solving across various topics, assessing scientific reasoning abilities. **TheoremQA** evaluates model's grasp and application of mathematical theorems, ranging from simple applications to complex proofs, testing advanced math skills.

Commonsense Reasoning (CR) This task evaluates model's ability to make deductions and understand everyday situations using implicit knowledge and logical inference. **TruthfulQA** (ThQA) evaluates model's ability to generate accurate and truthful responses, testing models on distinguishing fact from fiction, especially in areas prone to misconceptions. We use *BLEU* as the metric. **CoQA** assesses model's ability to understand and respond to questions in a conversational context, focusing on maintaining coherence and context throughout a dialogue. We use *F1 Score* as the metric.

Single Document QA (Single-Doc QA) This task assesses the model's reading comprehension skills when dealing with a single, extended document. **NarrativeQA** (Kociský et al., 2018) is a dataset designed to evaluate model's ability to comprehend and answer questions based on narrative texts, focusing on understanding stories and their underlying themes. **Qasper** (Dasigi et al., 2021) is a dataset aimed at assessing model's capability to extract and answer questions from academic papers, emphasizing understanding complex scientific information. We employ *F1 Score* as the metric for above two datasets.

Multi-Document QA (Multi-Doc QA) This task evaluates the model's reading comprehension capabilities across multiple extended documents. **2WikiMultiHopQA** (2WKMQA) (Ho et al., 2020) is a dataset designed to test model's ability to perform multi-hop reasoning and answer com-

	BS=1	BS=2		BS=8		BS=16		
[Input, Output]	[4K, 8K]	[8K, 16K]	[512, 4K]	[4K, 8K]	[512,512]	[4K, 4K]	[512, 512]	[2K, 2K]
Latency (s/100tokens)								
HF Accelerate	3.61	5.29	3.50	5.26	3.48	12.23	4.71	13.01
Ours	3.19	4.08	2.84	3.60	3.06	10.42	4.25	10.64
Throughput (token/s)								
HF Accelerate	27.71	18.89	57.08	38.01	230.00	65.41	339.43	122.98
ReFreeKV	31.35	24.51	70.30	55.56	261.83	76.81	376.89	150.40
Budget	73.3%	55.9%	62.0%	34.0%	76.7%	65.3%	82.8%	78.2%

Table 11: Performance comparison across various batch sizes and sequence lengths. We report latency (second/100 tokens, lower is better) and throughput (tokens/second, higher is better). More discussions regarding method efficiency are addressed in Section 4.3.

plex questions using information from multiple Wikipedia articles. **MuSiQue** (Trivedi et al., 2022) evaluates model's skill in integrating and reasoning over information from multiple sources to answer comprehensive questions accurately. We leverage *F1 Score* as the metric for above two datasets.

Summarization This task examines the model's ability to comprehend and summarize lengthy documents. **QMSum** (Zhong et al., 2021) is a dataset for evaluating model's ability to generate concise summaries of meeting transcripts, focusing on capturing the key points from multi-party discussions. **Multi-News** (M-News) (Fabbri et al., 2019) is a dataset that challenges models to create coherent summaries by synthesizing information from multiple news articles on the same topic. We use *Rouge-L* as the metric for above two datasets.

Few-Shot Learning (FSL) This task assesses the model's few-shot learning capabilities. **TriviaQA** (Joshi et al., 2017) is a dataset designed to assess model's ability to retrieve and answer questions based on large collections of trivia, emphasizing comprehension and factual recall. We use *F1 Score* as the metric.

Code This task evaluates the model's ability to complete and generate code. **LCC** (Guo et al., 2023) is a dataset focused on evaluating models' ability to understand and generate code by considering extended code contexts, enhancing the ability to reason over complex programming structures. We use *Edit Sim* as the metric.

D Ablation

In this section, we present additional ablation study results for Section 4.4. By setting various values for k in reducing the full attention matrix, we expand upon the results shown in Table 5, which are shown in Table 10. These experiments facilitate a deeper understanding of how different parameter settings impact model performance and provide a basis for optimizing parameter selection.

As shown in Table 10, setting $k = 1$ not only conserves pruning time but also achieves better model performance with a reduced budget, making it an ideal design choice.

E Comparison with StreamingLLM

We conduct additional experiments on six datasets using StreamingLLM (with Llama3-8B) while freezing the first two layers, as an ablation to assess the impact of the proposed two-stage process. As shown in Table 9, despite adopting the same layer-freezing strategy as ReFreeKV, StreamingLLM fails to achieve the objective of this work, as it cannot consistently maintain optimal performance across datasets such as GSM8K and GPQA under varying budgets. These results indicate that relying solely on the *position bias*, as in StreamingLLM, is insufficient to preserve full performance, and that incorporating additional design signals is necessary for achieving dynamic, lossless pruning.