# DropKV: Decoupling Residual-Output Perturbation for Near-Optimal KV-Cache Eviction

**Aozhong Zhang** [1] **Selcuk Gurses** [1] **Yanxia Deng** [2] **Naigang Wang** [3] **Chi-Chun Liu** [3] **Davis Wertheimer** [3]
**Derrick Liu** [3] **Xin Li** [4] **Zi Yang** [1] **Felix X.-F. Ye** [1] **Penghang Yin** [1]

## Abstract

Key-value (KV) cache management has become a critical bottleneck for scaling Large Language Models (LLMs) to long-context scenarios due to linear memory growth. While recent eviction methods have moved beyond simple attention-score heuristics to incorporate value representations, they typically face a fundamental trade-off: the underlying optimization for minimizing output perturbation is an NP-hard combinatorial problem, leading to mathematically suboptimal heuristics or implementations that are difficult to kernelize. In this work, we propose DropKV, a simple principled eviction framework based on **D**ecoupled **R**esidual-**O**utput **P**erturbation. By decoupling the joint eviction decision into independent per-token scoring, DropKV circumvents combinatorial intractability and admits a provable constant-factor approximation guarantee under a cone condition that we empirically verify on three long-context LLMs. To ensure practical viability, we provide a high-performance implementation using fused Triton kernels that avoid materializing the full attention matrix, delivering a 19.8× scoring-kernel speedup that translates to up to a 9.9% matched-batch end-to-end prefill speedup. Extensive evaluations on the RULER and LongBench benchmarks demonstrate that DropKV consistently outperforms state-of-the-art baselines at equivalent cache budgets, achieving the lowest inference latency among eviction methods while also remaining faster than the dynamic cache baseline.

[1]Department of Mathematics & Statistics, University at Albany, SUNY, Albany, NY, USA [2]Northwestern University, Evanston, IL, USA [3]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA [4]Department of Computer Science, University at Albany, SUNY, Albany, NY, USA. Correspondence to: Felix X.-F. Ye <xye2@albany.edu>, Penghang Yin <pyin@albany.edu>.

*Proceedings of the $43^{rd}$ International Conference on Machine Learning*, Seoul, South Korea. PMLR 306, 2026. 

## 1. Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Grattafiori et al., 2024; Jiang et al., 2023; Team et al., 2025; Yang et al., 2025b) are now deployed across code, scientific, and conversational tasks. As these models are increasingly used on long-form content such as entire codebases or lengthy transcripts, the underlying hardware efficiency becomes paramount. Autoregressive inference in transformer-based language models (Vaswani et al., 2017) relies heavily on key-value (KV) caching, which stores previously computed key and value projections to avoid redundant attention computation during the decoding phase. This mechanism effectively trades increased memory usage for significantly faster inference, making it a standard optimization in modern LLM inference systems. However, the KV cache memory requirement grows linearly with the sequence length and can become a prohibitive burden in long-context scenarios. For example, caching 128K tokens in a Phi-3-mini (Abdin et al., 2024) model requires approximately 48 GB of memory in `FP16` precision for the KV cache alone, which dwarfs the roughly 7 GB needed to store the whole model weights. This rapid expansion of the memory footprint leads to substantial GPU memory saturation and increased memory I/O overhead (Zhang et al., 2023), making KV cache a critical bottleneck for scalable long-context inference.

To address this challenge, a growing body of work has been proposed to reduce the KV cache memory footprint, including quantization (Hooper et al., 2024; Liu et al., 2024), low-rank compression (Chang et al., 2024; Sun et al., 2024), and eviction (Cai et al., 2024; Li et al., 2024; Park et al., 2025; Tang et al., 2024; Zhang et al., 2023). Among these methods, cache eviction has emerged as a practical solution that selectively retains only a small subset of tokens during inference. Existing eviction methods roughly fall into two categories. Query-agnostic methods evaluate importance based on intrinsic properties of the cache, independent of the current decoding state, such as token position (Xiao et al., 2023), the $\ell_2$-norms (Devoto et al., 2024) or pairwise similarities (Park et al., 2025) of the keys. These methods overlook the dynamic interaction between queries and cached keys. In contrast, query-aware methods leverage

attention signals to identify critical KV pairs (Li et al., 2024; Zhang et al., 2023). While more adaptive, existing methods typically equate importance solely with attention weights. This perspective is incomplete, as it ignores the critical role of the associated values; even a token with a moderate attention score can significantly shift the final representation if its corresponding value vector has a large magnitude or unique direction.

More recent methods have sought to transcend attention-score-only criteria by incorporating value representations into their selection logic (Feng et al., 2025; Geng et al., 2025) in a more principled manner. Specifically, they attempt to quantify the perturbation induced by cache eviction to the final attention output, in an effort to identify the optimal eviction set that minimizes the deviation from the original output. However, this underlying optimization is inherently an NP-hard combinatorial problem, forcing existing research to settle for mathematically suboptimal heuristics. Furthermore, as policy designs become increasingly intricate, it remains unclear whether these approaches can be efficiently kernelized, limiting their viability in production inference.

**Contributions.** In this work, we propose DropKV, a mathematically grounded and efficiently implemented KV-cache eviction framework. Specifically, our main contributions are as follows:

- **Decoupled Perturbation Framework:** We introduce a novel scoring mechanism that approximates the optimal eviction set by decoupling the NP-hard combinatorial search into a tractable per-token perturbation minimization with linear computational complexity.
- **Theoretical Near-Optimality Guarantees:** We prove that our decoupled proxy matches the optimal solution of the NP-hard problem up to a constant multiplicative factor under the cone condition, which we empirically verify on three long-context LLMs: Llama-3.1-8B, Mistral-7B-512K, and Qwen2.5-7B-1M.
- **High-Performance Triton Implementation:** We develop a suite of fused Triton kernels that compute eviction scores on-the-fly. By avoiding the materialization of the attention matrix, our implementation reduces the scoring-kernel scratch from $\sim$9 GB to $\sim$17 MB at $n$=128K, removing the prefill memory bottleneck and yielding end-to-end matched-batch prefill speedups.
- **State-of-the-Art Performance:** On RULER and LongBench at 5% and 10% cache budgets, DropKV achieves the highest accuracy and the lowest latency among query-aware and output-aware eviction baselines, while also remaining faster than the dynamic cache baseline.

## 2. Background and Related Work

### 2.1. Causal Attention and KV Cache

Transformer-based language models (Vaswani et al., 2017) rely on the self-attention mechanism to model long-range interactions among tokens in a sequence. Given an input matrix of hidden representations $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, the query, key, and value matrices are obtained via linear projections: $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \in \mathbb{R}^{n\times d}$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K \in \mathbb{R}^{n\times d}$, $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V \in \mathbb{R}^{n\times d}$, where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d\times d}$ are learnable projection matrices. The attention output is then computed as

$$\boldsymbol{A} := \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top + \boldsymbol{M}}{\sqrt{d}}\right)\boldsymbol{V} \in \mathbb{R}^{n\times d}, \qquad (1)$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a causal attention mask that prevents attending to future tokens. It is defined as $\boldsymbol{M}_{ij} = 0$ if $i \geq j$, and $-\infty$ otherwise. During autoregressive inference, the computation of self-attention proceeds in two phases: the prefill stage and the decoding stage.

In the *prefill* stage, the model processes the $n$-token prompt in a single forward pass via (1) and stores the resulting $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n\times d}$ in the KV cache. In the *decoding* stage, given the cached $\boldsymbol{K}_{1:t}, \boldsymbol{V}_{1:t}$, each new token at step $t{+}1$ is generated from its query $\boldsymbol{q}_{t+1}$ as $\boldsymbol{a}_{t+1}^\top = \mathrm{softmax}\left(\boldsymbol{q}_{t+1}^\top \boldsymbol{K}_{1:t}^\top/\sqrt{d}\right)\boldsymbol{V}_{1:t}$, after which its key $\boldsymbol{k}_{t+1}$ and value $\boldsymbol{v}_{t+1}$ are appended to the cache.

### 2.2. KV-Cache Eviction

The challenge of efficient long-context inference has led to a variety of strategies for KV cache eviction. Early approaches relied on simple sliding windows (Beltagy et al., 2020) to maintain recent context, though these often risked losing critical historical information. To address this, StreamingLLM (Xiao et al., 2023) introduced "attention sinks", demonstrating that preserving a small set of initial tokens is helpful for maintaining the functionality of the attention mechanism. KNorm (Devoto et al., 2024) retains tokens with smaller key norms, while KeyDiff (Park et al., 2025) prefers tokens whose keys are more diverse in the representation space. KVzip (Kim et al., 2025) evaluates KV cache importance by prompting the model to reconstruct chunked contexts and aggregating the maximum attention weights received by the context keys. Methods like H2O (Zhang et al., 2023) and Scissorhands (Liu et al., 2023c) moved toward importance-based selection by utilizing accumulated attention scores to identify "heavy hitter" entries. This line of research culminated in state-of-the-art frameworks like SnapKV (Li et al., 2024), which employs observation window-based accumulation and pooling operations to consolidate attention weights. More recently, the focus has expanded from token-level scoring to architec-

tural optimization; for instance, AdaKV (Feng et al., 2024) and related budget-allocation methods (Cai et al., 2024; Qin et al., 2025; Xiao et al., 2024; Yang et al., 2024) dynamically redistribute cache capacity across different attention heads to better capture their unique characteristics. Despite these advancements, existing strategies remain largely empirical, relying almost exclusively on attention weights as a proxy for token importance.

Newer inquiries have begun to bridge this gap by incorporating value representations into the eviction logic (Feng et al., 2025; Geng et al., 2025). These methods can be viewed as *output-aware* eviction strategies that attempt to indirectly minimize the attention-output perturbation induced by token removal. For example, CriticalKV (Feng et al., 2025) seeks to minimize a loose upper-bound on the attention-output perturbation, while AnDPro (Geng et al., 2025) formulates the combinatorial perturbation minimization problem through a convex relaxation. DropKV instead scores tokens using the exact per-token output perturbation and further admits a constant-factor approximation guarantee for the original joint multi-token eviction problem under a cone condition (§4). See Appendix B for detailed comparisons with existing methods.

A separate line of work targets the same memory/bandwidth bottleneck without permanently removing tokens: dynamic sparse attention keeps the full KV cache and attends to only a subset per step (Quest (Tang et al., 2024), SparQ (Ribar et al., 2023), Loki (Singhania et al., 2024)), while SVD-based compression and offload methods retain all token positions but compress or stream the cache between GPU and CPU (ShadowKV (Sun et al., 2024), xKV (Chang et al., 2025), InfiniGen (Lee et al., 2024)). These approaches are orthogonal to the eviction track, in which DropKV operates: tokens are permanently removed and the resident cache footprint shrinks.

## 3. Method

### 3.1. Attention-Output Perturbation

Recall that in an autoregressive transformer with $n$ input tokens, the attention output $\boldsymbol{a}$ for a query vector $\boldsymbol{q} \in \mathbb{R}^d$ is computed as $\boldsymbol{p} = \mathrm{softmax}\big(\boldsymbol{K}\boldsymbol{q}/\sqrt{d}\big) \in \mathbb{R}^n$, $\boldsymbol{a} = \boldsymbol{V}^\top \boldsymbol{p} = \sum_{i=1}^n p_i \boldsymbol{v}_i \in \mathbb{R}^d$, where $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n\times d}$ are the accumulated key and value projection matrices in the KV cache, $\boldsymbol{v}_i^\top$ is the $i$-th row vector of $\boldsymbol{V}$, and $p_i \in (0,1)$ is the $i$-th attention weight after softmax, satisfying $\sum_{i=1}^n p_i = 1$. The following result quantifies the perturbation to the attention output, $\Delta(\mathcal{J})$, associated with query $\boldsymbol{q}$ induced by an index set $\mathcal{J}$ of evicted tokens. That is, the perturbation depends not only on the attention scores but also on the value residuals of the form $\boldsymbol{a} - \boldsymbol{v}_i$.

**Lemma 1** (Perturbation)**.** *Suppose a set of tokens indexed by $\mathcal{J} \subset [n]$ is evicted from the full KV cache, and let $\boldsymbol{a}_{-\mathcal{J}}$ denote the resulting attention output. Then the post-eviction shift $\Delta(\mathcal{J}) := \boldsymbol{a}_{-\mathcal{J}} - \boldsymbol{a}$ associated with the query $\boldsymbol{q}$ satisfies*

$$\Delta(\mathcal{J}) = \frac{\sum_{i\in\mathcal{J}} p_i(\boldsymbol{a} - \boldsymbol{v}_i)}{1 - P_{\mathcal{J}}},$$

*where $P_{\mathcal{J}} := \sum_{i\in\mathcal{J}} p_i$ denotes the total attention weights associated with the evicted tokens. In particular, for $\mathcal{J} = \{j\}$, we have $\Delta(\{j\}) = \frac{p_j}{1-p_j}(\boldsymbol{a} - \boldsymbol{v}_j)$ for per-token eviction.*

Crucially, all quantities necessary for evaluating $\Delta(\mathcal{J})$ can be precomputed *prior to the execution of any eviction decision*. Equivalent formulations of $\Delta(\mathcal{J})$ in the multi-token case have appeared in recent studies CriticalKV (Feng et al., 2025) and AnDPro (Geng et al., 2025). Unlike the explicit expression in Lemma 1, these works introduce binary variables to model token selection and derive eviction criteria by relaxing the resulting multi-token perturbation objective. In contrast, we directly adopt the exact per-token perturbation and show that this decoupled criterion is near-optimal under mild conditions. Detailed comparisons are provided in § B. For completeness, we include the proof of Lemma 1 in § C.

### 3.2. Combinatorial Objective and a Decoupled Proxy

To determine the optimal eviction set $\mathcal{J}^*$ of $k$ tokens, one seeks to solve the following combinatorial optimization problem:

$$\mathcal{J}^* = \arg\min_{\substack{\mathcal{J}\subset[n]\\ |\mathcal{J}|=k}} \|\Delta(\mathcal{J})\|_2, \qquad \text{(Exact)}$$

where $\|\Delta(\mathcal{J})\|_2 = \frac{\|\sum_{i\in\mathcal{J}} p_i(\boldsymbol{a}-\boldsymbol{v}_i)\|_2}{1-P_{\mathcal{J}}}$. However, this optimization is NP-hard (Appendix C.1): brute-force enumeration of the $\binom{n}{k}$ feasible subsets is infeasible even for moderate $n$ and $k$, motivating a tractable proxy.

To circumvent this intractability, we propose a simple surrogate: select the $k$ tokens that individually induce the smallest output perturbation. Formally, we perform a smallest topk selection, denoted by $\mathrm{topk}^-$, over the per-token perturbation loss:

$$\hat{\mathcal{J}} = \arg\mathrm{topk}^-_{j\in[n]}\left(\|\Delta(\{j\})\|_2, k\right), \qquad \text{(Proxy)}$$

where $\|\Delta(\{j\})\|_2 = \frac{p_j}{1-p_j}\big\|\boldsymbol{a} - \boldsymbol{v}_j\big\|_2$ is the perturbation norm from evicting only the $j$-th token. Since each per-token perturbation is independent, all $n$ scores cost $O(nd)$ in parallel, making (Proxy) dramatically more tractable than the combinatorial problem in (Exact).

Our proposed proxy decouples the joint eviction decision into independent per-token scoring by simply omitting the interactions among evicted tokens. While this independence assumption might seem crude, we prove theoretically in § 4

that it is surprisingly *near-optimal* under conditions commonly observed in practice. Specifically, when the attention weights $p_j$ are small (Assumption 1) and the residual directions $\boldsymbol{a} - \boldsymbol{v}_j$ are approximately aligned (Assumption 2), the proxy $\hat{\mathcal{J}}$ closely approximates the optimal eviction policy $\mathcal{J}^*$ in terms of perturbation loss (Theorem 1).

### 3.3. Multi-Query Aggregation

For each token $j$, we define the importance score based on the cumulative eviction loss over an observation window of queries $\mathcal{Q} = \{\boldsymbol{q}^{(t)}\}_{t=1}^{w}$, similar to SnapKV (Li et al., 2024). Specifically, accounting for the dependence of the attention weight $p_j^{(t)}$ and output $\boldsymbol{a}^{(t)}$ on the query $\boldsymbol{q}^{(t)}$, we score the $j$-th token by total squared perturbation error across $\mathcal{Q}$:

$$\begin{aligned}\text{score}_j &:= \sum_{t=1}^{w} \|\Delta^{(t)}(\{j\})\|_2^2 \\ &= \sum_{t=1}^{w} \left(\frac{p_j^{(t)}}{1 - p_j^{(t)}}\right)^2 \|\boldsymbol{a}^{(t)} - \boldsymbol{v}_j\|_2^2\end{aligned}$$

The squared form is a kernel-friendly surrogate: it enables the fused-tile implementation described in § D.1, while at $w{=}1$ preserving the same $\text{topk}^-$ ranking as the unsquared proxy (Proxy). Theorem 1 establishes the constant-factor bound for the single-query proxy (Proxy); the multi-query case ($w > 1$) follows the standard observation-window practice (Feng et al., 2025; Li et al., 2024). Finally, tokens are ranked by their scores, and the $k$ tokens with the smallest scores are selected for eviction:

$$\text{topk}^-_{j\in[n]}\left(\text{score}_j, k\right). \tag{DropKV}$$

The pseudocode for DropKV is summarized in Algorithm 1.

## 4. Theoretical Guarantees

In this section, we establish formal guarantees for our proxy (Proxy). To make the analysis tractable and realistic, we restrict attention to eviction over a *candidate pool* $\mathcal{L} \subseteq [n]$ (e.g., the set of tokens with the lowest attention scores) and consider only subsets $\mathcal{J} \subseteq \mathcal{L}$ with exactly $|\mathcal{J}| = k$. We begin with two assumptions and report their numerical verification in Appendix C.3.

**Assumption 1** (Small Eviction Probabilities)**.** *There exists $\varepsilon \in (0, 1)$ such that $p_i \le \varepsilon$ for all $i \in \mathcal{L}$.*

Under this condition, any set $\mathcal{J} \subseteq \mathcal{L}$ with $|\mathcal{J}| = k$ satisfies $P_{\mathcal{J}} = \sum_{i\in\mathcal{J}} p_i \le k\varepsilon$. When $\varepsilon$ is small (as is typical for low-attention tokens selected for eviction), $P_{\mathcal{J}}$ remains negligible. Next, we impose a directional alignment condition on the residual vectors $\boldsymbol{r}_i := \boldsymbol{a} - \boldsymbol{v}_i \in \mathbb{R}^d$ for $i \in \mathcal{L}$.

**Assumption 2** (Cone Condition)**.** *There exist a unit vector $\boldsymbol{u} \in \mathbb{R}^d$ with $\|\boldsymbol{u}\|_2 = 1$ and a constant $c \in (0, 1]$ such that*

$$\boldsymbol{u}^\top \boldsymbol{r}_i \ge c\|\boldsymbol{r}_i\|_2, \quad \forall\, i \in \mathcal{L}.$$

Intuitively, the cone condition requires the perturbations induced by different token removals to be sufficiently aligned rather than adversarially canceling each other. Our main result, Theorem 1, shows that, under these assumptions, restricting the top-k selection in (Proxy) to the candidate pool $\mathcal{L}$ yields a near-optimal objective value in (Exact). It provides an approximation matching the optimal solution of the NP-hard problem (Exact) up to a constant multiplicative factor. The full proof is deferred to Appendix C.

**Theorem 1** (Near-Optimality)**.** *Let $\mathcal{J}^* = \arg\min\{F(\mathcal{J}) := \|\Delta(\mathcal{J})\|_2 : \mathcal{J} \subseteq \mathcal{L},\ |\mathcal{J}| = k\}$ be the optimal eviction set and $\hat{\mathcal{J}} = \arg\text{topk}^-_{j\in\mathcal{L}}\left(\|\Delta(\{j\})\|_2, k\right)$ the proxy. Under Assumptions 1 and 2, we have*

$$F(\hat{\mathcal{J}}) \le \frac{1}{c(1-\varepsilon)(1-k\varepsilon)}\, F(\mathcal{J}^*).$$

## 5. Experiments

We evaluate our method on multiple long-context language models, benchmarks, and competitive baselines to examine its effectiveness across different architectures and context regimes.

**Models and Baselines.** We evaluate on three representative decoder-only LLMs: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), MegaBeam-Mistral-7B-512K (Wu & Song, 2025) (Mistral-7B-512K), and Qwen2.5-7B-Instruct-1M (Yang et al., 2025a). Mistral-7B-512K is a long-context extension of Mistral-7B-Instruct-v0.2 that supports context lengths up to 512K tokens, and Qwen2.5-7B-Instruct-1M is the long-context version of Qwen2.5 with a maximum context length of 1M tokens. We compare against a broad set of competitive baselines, including SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), StreamingLLM (Xiao et al., 2023), KeyDiff (Park et al., 2025), CAKE (Qin et al., 2025), CriticalKV (Feng et al., 2025), and AnDPro (Geng et al., 2025). For each method, we follow the default settings in the original paper whenever possible and only adjust the cache budget to enable matched-budget comparison.[1]

**Benchmarks.** We use two widely adopted long-context benchmarks: RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2024). RULER is a synthetic benchmark with configurable sequence lengths and task complexity for evaluating long-context capabilities beyond simple in-context recall. LongBench is a multitask benchmark covering single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion, with automatic evaluation for scalable assessment. We use the 16 English and code tasks from LongBench. The full task lists are provided in Appendix E.

[1] Implementations of several baselines are adapted from the public codebase: https://github.com/NVIDIA/kvpress.

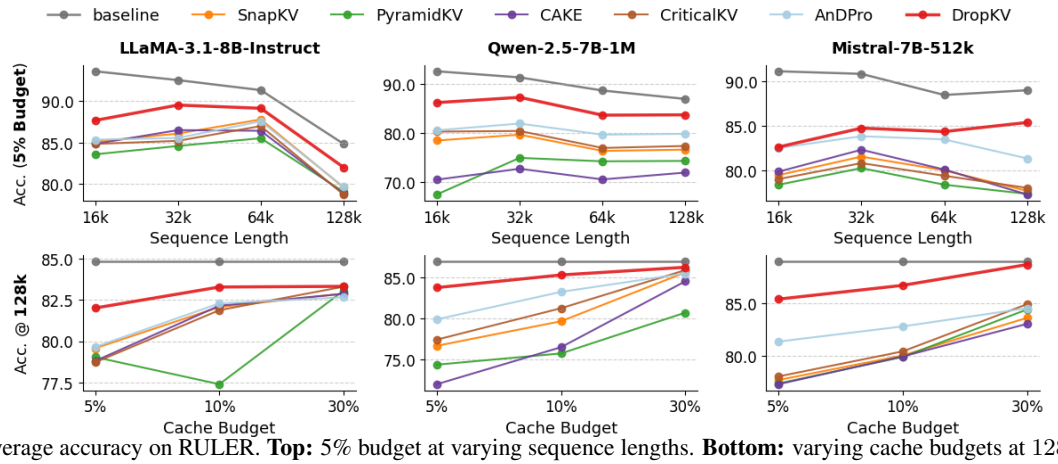



*Figure 1.* Average accuracy on RULER. **Top:** 5% budget at varying sequence lengths. **Bottom:** varying cache budgets at 128K sequence length. Our method consistently outperforms baselines and shows more graceful degradation at longer contexts.

*Table 1.* LongBench results at 5% KV-cache budget.

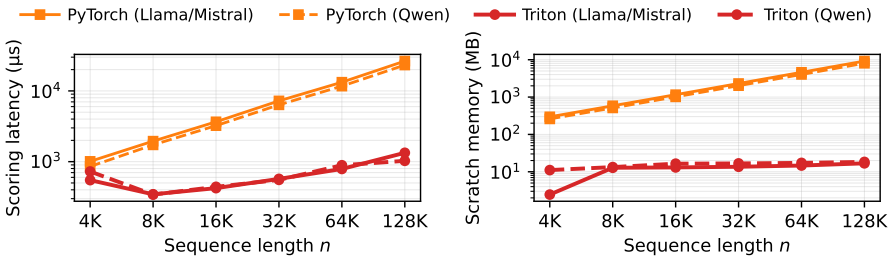
| Method | Single Doc. QA | | | Multi Doc. QA | | | Summarization | | | Synthetic | | Fewshot Learning | | | Code | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NrtvQA | Qasper | MF-en | HotpotQA | 2WikiMQA | Musique | GovReport | QMSum | MultiNews | PCount | PR-en | TREC | TriviaQA | SAMSum | Lcc | RB-P | |
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | | | | | |
| FullKV | 30.51 | 45.59 | 56.34 | 58.22 | 50.02 | 31.75 | 34.3 | 25.33 | 26.76 | 9.5 | 100 | 73.5 | 92.29 | 43.96 | 64.77 | 55.7 | 49.91 |
| SnapKV | 30.34 | 31.51 | 54.45 | 56.95 | 47.34 | 29.8 | 22.64 | 24.62 | 16.03 | 9.5 | 100 | 39 | 92.03 | 42.18 | 58.82 | 54.97 | 44.39 |
| PyramidKV | 31.08 | 31.42 | 54.37 | 56.92 | 47.34 | 30.62 | 22.4 | 24.27 | 16.33 | 9.5 | 100 | 39.5 | 91.92 | 42.46 | 59.44 | 54.27 | 44.49 |
| StreamingLLM | 25.91 | 21.94 | 33.16 | 49.06 | 41.35 | 23.91 | 21.67 | 21.5 | 16.86 | 9.5 | 97 | 26.5 | 84.55 | 41.9 | 58.02 | 49.83 | 38.92 |
| KeyDiff | 27.93 | 27.17 | 38.52 | 51.92 | 36.8 | 26.54 | 23.02 | 22.91 | 18.31 | 9.5 | 63.5 | 30.5 | 78.52 | 22.36 | 32.21 | 31.65 | 33.84 |
| CAKE | 30.41 | 33.54 | 54.79 | 57.09 | 48.29 | 31.07 | 23.54 | 24.36 | 17.33 | 9.5 | 100 | 37 | 91.05 | 42.54 | 58.58 | 53.7 | 44.55 |
| CriticalKV | 30.03 | 31.07 | 54.55 | 57.33 | 47.15 | 30.09 | 23.16 | 23.92 | 18.46 | 9.5 | 100 | 41 | 92.03 | 42.14 | 59.25 | 54.54 | 44.64 |
| AnDPro | 30.97 | 32.06 | 54.44 | 57.62 | 46.44 | 31.53 | 22.92 | 24.25 | 17.98 | 9.25 | 100 | 38 | 91.65 | 41.86 | 58.58 | 54.9 | 44.53 |
| DropKV | 30.48 | 32.49 | 55.34 | 57.29 | 46.84 | 31.25 | 23.71 | 24.94 | 18.7 | 9.5 | 100 | 48.5 | 92.27 | 43.06 | 60.29 | 53.47 | **45.51** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | | | | | |
| FullKV | 28.88 | 48.39 | 51.57 | 60.27 | 53.19 | 35.63 | 33.33 | 23.22 | 24.53 | 6.5 | 100 | 66.5 | 85.28 | 45.29 | 45.61 | 35.31 | 46.47 |
| SnapKV | 27.76 | 35.9 | 48.74 | 59.15 | 51.51 | 34.19 | 23.42 | 22.32 | 13.13 | 6.5 | 100 | 33 | 84.66 | 44.2 | 38.75 | 34.47 | 41.11 |
| PyramidKV | 27.19 | 34.45 | 48.49 | 57.01 | 51.57 | 32.37 | 22.67 | 21.5 | 13.09 | 5.5 | 99 | 33.5 | 84.93 | 43.55 | 38.46 | 33.07 | 40.40 |
| StreamingLLM | 21.48 | 28.86 | 27.5 | 47.08 | 42.58 | 22.99 | 21.86 | 18.51 | 13.97 | 6.5 | 16.5 | 27 | 80.62 | 42.26 | 37.84 | 32.34 | 30.49 |
| KeyDiff | 20.48 | 24.5 | 21.94 | 38.44 | 25.67 | 17.35 | 21.19 | 17.17 | 12.11 | 6.5 | 13 | 19.5 | 42.65 | 13.11 | 20.75 | 18.96 | 20.83 |
| CAKE | 28.75 | 38.51 | 49.22 | 59.59 | 52.14 | 33.85 | 24.12 | 22.59 | 14.08 | 7 | 100 | 36 | 84.71 | 44.35 | 37.53 | 35.06 | 41.72 |
| CriticalKV | 28.31 | 38.22 | 48.83 | 57.63 | 51.99 | 34.11 | 24.4 | 22.35 | 14.83 | 7 | 100 | 36.5 | 84.61 | 43.83 | 38.59 | 34.73 | 41.52 |
| AnDPro | 28.61 | 36.11 | 49.88 | 59.44 | 50.87 | 34.56 | 23.61 | 22.24 | 14.22 | 6.5 | 100 | 37.5 | 84.71 | 44.21 | 38.08 | 34.76 | 41.58 |
| DropKV | 28.28 | 39.09 | 50.54 | 58.87 | 51.6 | 33.93 | 25.25 | 21.42 | 15.36 | 7 | 100 | 48.5 | 85.06 | 44.85 | 38.95 | 33.88 | **42.66** |
| **Mistral-7B-512K** | | | | | | | | | | | | | | | | | |
| FullKV | 22.04 | 16.47 | 42.75 | 24.06 | 20.72 | 13.04 | 33.35 | 25.25 | 26.97 | 0 | 100 | 74.5 | 89.55 | 41.69 | 55.54 | 56.39 | 40.15 |
| SnapKV | 21.39 | 10.98 | 41.92 | 18.68 | 16.25 | 9.21 | 23.38 | 24.53 | 17.4 | 0 | 100 | 38.5 | 90.3 | 39.23 | 50.64 | 53.6 | 34.75 |
| PyramidKV | 20.4 | 11.25 | 42.22 | 19.62 | 15.67 | 8.3 | 23.34 | 24.74 | 17.18 | 0 | 100 | 38.5 | 90.13 | 39.09 | 50.67 | 52.81 | 34.62 |
| StreamingLLM | 18.25 | 9.28 | 25.16 | 14.07 | 13.96 | 5.92 | 18.79 | 21.41 | 16.65 | 0.5 | 20.5 | 22.5 | 86.81 | 38.74 | 47.26 | 49.87 | 25.60 |
| KeyDiff | 19.49 | 10.43 | 22.08 | 18.99 | 14.74 | 7.25 | 22.09 | 21.76 | 19.13 | 0.5 | 6.5 | 16 | 58.84 | 17.53 | 24.25 | 23.84 | 18.96 |
| CAKE | 21.3 | 11.7 | 43.07 | 18.54 | 14.95 | 11.11 | 24.3 | 24.67 | 19.67 | 0 | 100 | 46 | 89.39 | 39.62 | 50.1 | 54.25 | 35.54 |
| CriticalKV | 21.19 | 13.72 | 42.78 | 17.72 | 15.83 | 9.89 | 24.15 | 24.18 | 20.12 | 0 | 100 | 43 | 90.08 | 40.29 | 51.77 | 53.88 | 35.54 |
| AnDPro | 19.93 | 12.96 | 40.49 | 19.01 | 14.17 | 9.74 | 23.57 | 24.51 | 19.11 | 0.5 | 100 | 44.5 | 90.3 | 39.23 | 49.36 | 53.23 | 35.04 |
| DropKV | 21.52 | 11.64 | 41 | 17.6 | 14.39 | 10.58 | 25.23 | 24.98 | 20.13 | 0.5 | 100 | 57 | 90.3 | 39.99 | 52.76 | 51.9 | **36.22** |

**Implementation details.** We compare all methods and report results at **5%**, **10%** and **30%** KV-cache budgets. Following SnapKV (Li et al., 2024), we use the last 8 query tokens and set the pooling kernel size to 11 when computing the eviction scores. All experiments are conducted on an NVIDIA A100-80GB GPU and evaluated within a unified lm-evaluation-harness (Gao et al., 2024) pipeline to ensure a consistent and fair comparison across methods. More Triton implementation details are provided in Appendix D.

### 5.1. RULER

We evaluate all methods on the RULER across sequence lengths from 16K to 128K. Fig. 1 shows accuracy across sequence lengths at 5% budget (top) and across cache budgets at 128K (bottom). Overall, our method consistently achieves the best performance across all three model families and remains the closest to the Full-KV baseline.

Fig. 1 further shows that the advantage of our method is consistent across context lengths. In particular, under high compression, competing methods exhibit a much sharper degradation as the sequence length increases, whereas our method degrades more gracefully and consistently outperforms prior approaches. This trend is especially clear at 128K, where our method preserves accuracy significantly better across all three models. All RULER results are provided in Appendix A, including full per-task breakdowns as well as evaluations across different cache budget and sequence lengths; See Table 5–8.




| | 4K | 32K | 128K |
|---|---|---|---|
| Llama | +7.2 | +9.9 | +2.5 |
| Mistral | +7.0 | +4.9 | +2.7 |
| Qwen | +1.9 | −0.1 | +3.3 |

*Figure 2.* DropKV scoring at $B$=1, $w$=8, and 5% budget. **Left:** Triton vs. PyTorch on scoring latency and peak scratch memory. At $n$=128K Triton is ∼20× faster with scratch flat at ∼17 MB vs. ∼9 GB. **Right:** Matched-batch prefill relative speedup percentage is positive in 8 of 9 cells; Qwen at $n$=32K is essentially neutral (−0.1%).

### 5.2. LongBench

Table 1 reports the LongBench results at 5% KV-cache budget. Our method achieves the highest average score on all three model families, reaching 45.51 on Llama-3.1-8B-Instruct, 42.66 on Qwen2.5-7B-1M, and 36.22 on Mistral-7B-512K. Compared with the strongest baseline, this gives improvements of 0.87, 0.94, and 0.68 points, respectively.

These results show that our method remains effective across a diverse set of LongBench tasks under aggressive compression ratio. In addition to the gains in the overall average, our method also performs competitively on multiple task categories. In the main paper, we only report the results at 5% budget in Table 1; additional results are provided in Appendix A (Table 3 & 4).

### 5.3. Memory and Latency

We evaluate the Triton path against a matched PyTorch reference on the three long-context LLMs on an A100-80GB.

**Isolated scoring** (Fig. 2 left). At $n$=128K, $B$=1, $w$=8, and 5% budget the fused Triton kernel is 19.8× faster than the PyTorch reference while holding scoring-kernel scratch flat at ∼17 MB vs. ∼9 GB (a ∼540× scratch reduction; total peak allocation drops from 10.6 GB to 1.6 GB on the heaviest of the three target models).

**End-to-end prefill at matched batch size** (Fig. 2 right). At the lower of the two methods' max batch per cell, matched-batch prefill speedup $\delta := (t_{\text{py}} - t_{\text{tri}})/t_{\text{tri}}$ is positive in eight of nine $(n, \text{model})$ cells of $n \in \{4\text{K}, 32\text{K}, 128\text{K}\}$, ranging +1.9% to +9.9%; the lone exception is Qwen at $n$=32K ($\delta$=−0.1%). The dynamic cache (no eviction) admits a strictly smaller batch than DropKV at every $n$ and model, so a matched-batch comparison with dynamic is not defined; we instead compare each method at its own capacity ceiling below.

**Serving throughput vs. dynamic cache.** Decoding throughput at each method's maximum feasible batch is 1.75–8.22× higher than the dynamic cache at 5% budget (Table 9 in Appendix A). This gain comes from the KV-bandwidth reduction shared by any eviction method at this budget: per-step decode latency is bandwidth-bound by the (post-eviction) KV size and is therefore comparable across eviction methods. Triton's contribution is concentrated in prefill (the 19.8× scoring speedup and ∼540× scratch reduction reported above), not per-step decode; Triton vs. PyTorch differences in $B$ within Table 9 reflect the ∼9 GB prefill-scratch reduction crossing the batch-fit boundary in some cells (e.g., Mistral $n$=32K: $B$=12 vs. 11).

*Table 2.* End-to-End latencies (in seconds) across batch sizes $B \in \{1, 2, 4, 8, 12\}$. All evaluations were conducted using the Mistral-7B-512K model with 32K sequence length. ↓ indicates lower values are better. "OOM" denotes Out-of-Memory.

| Method | End-to-End Latency (s) ↓ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 12 |
| Dynamic | 19.08 | 30.91 | 54.72 | OOM | OOM |
| SnapKV | 16.44 | 20.22 | 27.71 | 43.08 | 59.10 |
| CriticalKV | 16.72 | 21.10 | 29.55 | 46.36 | 64.75 |
| AnDPro | 17.79 | 23.06 | 33.22 | 53.91 | 76.25 |
| **DropKV** | **16.21** | **20.13** | **27.50** | **42.71** | **59.01** |

**End-to-end full generation latency.** Table 2 reports full-generation latency on Mistral-7B-512K with 32K seq. length, 500 decoding steps, 5% budget. DropKV achieves the lowest full-generation latency across all batch sizes; the dynamic cache runs out of memory beyond $B$=4, while all eviction methods scale to $B$=12. Among eviction methods, DropKV consistently delivers the lowest total generation time at every batch size. Additional time-to-first-token results are provided in Appendix A.1.

## 6. Discussion and Limitation

In this work, we introduced DropKV, a framework that bridges the gap between theoretical optimality and practical scalability in KV cache eviction. By decoupling the joint perturbation problem, we demonstrated that near-optimal cache eviction can be achieved with minimal computational overhead. Our empirical results suggest that the cone condition, defined as the alignment of value residuals, holds at the majority of (layer, head, query-position) triples in our three target long-context LLMs, providing a mathematical justification for why independent per-token scoring remains effective. Our approach uses the same cache budget across

the attention layers/heads. However, different attention layers/heads and decoding steps may exhibit varying sensitivity to information loss. In future work, we plan to extend DropKV in two directions: an adaptive per-layer/per-head budget that distributes capacity according to head sensitivity, and a Triton decode-time kernel, since our current implementation accelerates prefill scoring only.

## Acknowledgement

This work was in part supported by NSF grant DMS-2208126, the UAlbany-IBM CEAIS seed grant, and the IBM PhD Fellowship Award.

## References

Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., Benhaim, A., Bilenko, M., Bjorck, J., Bubeck, S., Cai, M., Cai, Q., Chaudhary, V., Chen, D., Chen, D., Chen, W., Chen, Y.-C., Chen, Y.-L., Cheng, H., Chopra, P., Dai, X., Dixon, M., Eldan, R., Fragoso, V., Gao, J., Gao, M., Gao, M., Garg, A., Giorno, A. D., Goswami, A., Gunasekar, S., Haider, E., Hao, J., Hewett, R. J., Hu, W., Huynh, J., Iter, D., Jacobs, S. A., Javaheripi, M., Jin, X., Karampatziakis, N., Kauffmann, P., Khademi, M., Kim, D., Kim, Y. J., Kurilenko, L., Lee, J. R., Lee, Y. T., Li, Y., Li, Y., Liang, C., Liden, L., Lin, X., Lin, Z., Liu, C., Liu, L., Liu, M., Liu, W., Liu, X., Luo, C., Madan, P., Mahmoudzadeh, A., Majercak, D., Mazzola, M., Mendes, C. C. T., Mitra, A., Modi, H., Nguyen, A., Norick, B., Patra, B., Perez-Becker, D., Portet, T., Pryzant, R., Qin, H., Radmilac, M., Ren, L., de Rosa, G., Rosset, C., Roy, S., Ruwase, O., Saarikivi, O., Saied, A., Salim, A., Santacroce, M., Shah, S., Shang, N., Sharma, H., Shen, Y., Shukla, S., Song, X., Tanaka, M., Tupini, A., Vaddamanu, P., Wang, C., Wang, G., Wang, L., Wang, S., Wang, X., Wang, Y., Ward, R., Wen, W., Witte, P., Wu, H., Wu, X., Wyatt, M., Xiao, B., Xu, C., Xu, J., Xu, W., Xue, J., Yadav, S., Yang, F., Yang, J., Yang, Y., Yang, Z., Yu, D., Yuan, L., Zhang, C., Zhang, C., Zhang, J., Zhang, L. L., Zhang, Y., Zhang, Y., Zhang, Y., and Zhou, X. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Cai, Z., Zhang, Y., Gao, B., Liu, Y., Li, Y., Liu, T., Lu, K., Xiong, W., Dong, Y., Hu, J., et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Chang, C.-C., Lin, W.-C., Lin, C.-Y., Chen, C.-Y., Hu, Y.-F., Wang, P.-S., Huang, N.-C., Ceze, L., Abdelfattah, M. S., and Wu, K.-C. Palu: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

Chang, C.-C., Lin, C.-Y., Akhauri, Y., Lin, W.-C., Wu, K.-C., Ceze, L., and Abdelfattah, M. S. xkv: Cross-layer svd for kv-cache compression. *arXiv preprint arXiv:2503.18893*, 2025.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning, 2023.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*, 2021.

Devoto, A., Zhao, Y., Scardapane, S., and Minervini, P. A simple and effective l_2 norm-based strategy for kv cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18476–18499, 2024.

Fabbri, A. R., Li, I., She, T., Li, S., and Radev, D. R. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*, 2019.

Feng, Y., Lv, J., Cao, Y., Xie, X., and Zhou, S. K. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*, 2024.

Feng, Y., Lv, J., Cao, Y., Xie, X., and Zhou, S. K. Identify critical kv cache in llm inference from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*, 2025.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Geng, Z., Wang, J., Liu, Z., Ju, F., Li, Y., Li, X., Yuan, M., Hao, J., Lian, D., Chen, E., et al. Accurate kv cache eviction via anchor direction projection for efficient llm inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Gliwa, B., Mochol, I., Biesek, M., and Wawer, A. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*, 2019.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Guo, D., Xu, C., Duan, N., Yin, J., and McAuley, J. Longcoder: A long-range pre-trained language model for code completion, 2023. URL https://arxiv.org/abs/2306.14893.

Hartmanis, J. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.

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

Ho, X., Nguyen, A.-K. D., Sugawara, S., and Aizawa, A. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020b.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*, 2021.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

Kamradt., G. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

Kim, J.-H., Kim, J., Kwon, S., Lee, J. W., Yun, S., and Song, H. O. Kvzip: Query-agnostic kv cache compression with context reconstruction. *arXiv preprint arXiv:2505.23416*, 2025.

Lee, W., Lee, J., Seo, J., and Sim, J. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 155–172, 2024.

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

Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023a.

Liu, T., Xu, C., and McAuley, J. Repobench: Benchmarking repository-level code auto-completion systems, 2023b. URL https://arxiv.org/abs/2306.03091.

Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023c.

Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

Park, J., Jones, D., Morse, M. J., Goel, R., Lee, M., and Lott, C. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *arXiv preprint arXiv:2504.15364*, 2025.

Qin, Z., Cao, Y., Lin, M., Hu, W., Fan, S., Cheng, K., Lin, W., and Li, J. Cake: Cascading and adaptive kv cache eviction with layer preferences. *arXiv preprint arXiv:2503.12491*, 2025.

Ribar, L., Chelombiev, I., Hudlass-Galley, L., Blake, C., Luschi, C., and Orr, D. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*, 2023.

Singhania, P., Singh, S., He, S., Feizi, S., and Bhatele, A. Loki: Low-rank keys for efficient sparse attention. *Advances in Neural Information Processing Systems*, 37: 16692–16723, 2024.

Sun, H., Chang, L.-W., Bao, W., Zheng, S., Zheng, N., Liu, X., Dong, H., Chi, Y., and Chen, B. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., and Han, S. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., yeong Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, C. and Song, Y. Scaling context, not parameters: Training a compact 7b language model for efficient long-context processing. *arXiv preprint arXiv:2505.08651*, 2025.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Xiao, G., Tang, J., Zuo, J., Guo, J., Yang, S., Tang, H., Fu, Y., and Han, S. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.

Yang, A., Yu, B., Li, C., Liu, D., Huang, F., Huang, H., Jiang, J., Tu, J., Zhang, J., Zhou, J., Lin, J., Dang, K., Yang, K., Yu, L., Li, M., Sun, M., Zhu, Q., Men, R., He, T., Xu, W., Yin, W., Yu, W., Qiu, X., Ren, X., Yang, X.,

Li, Y., Xu, Z., and Zhang, Z. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025a.

Yang, D., Han, X., Gao, Y., Hu, Y., Zhang, S., and Zhao, H. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3258–3270, 2024.

Yang, Q. A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*, 2021.

# A. Additional Experimental Results: Longbench & RULER

Tables 3 and 4 extend the LongBench evaluation in §5 to 10% and 30% cache budgets.

*Table 3.* LongBench results at 10% cache budget.

| | Single Doc. QA | | | Multi Doc. QA | | | Summarization | | | Synthetic | | Fewshot Learning | | | Code | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | NrtvQA | Qasper | MF-en | HotpotQA | 2WikiMQA | Musique | GovReport | QMSum | MultiNews | PCount | PR-en | TREC | TriviaQA | SAMSum | Lcc | RB-P | Avg. |
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | | | | | |
| FullKV | 30.51 | 45.59 | 56.34 | 58.22 | 50.02 | 31.75 | 34.3 | 25.33 | 26.76 | 9.5 | 100 | 73.5 | 92.29 | 43.96 | 64.77 | 55.7 | 49.91 |
| SnapKV | 30.11 | 37.66 | 55.52 | 57.27 | 48.37 | 31.51 | 25.77 | 24.24 | 19.55 | 9.5 | 100 | 49.5 | 91.54 | 41.68 | 62.02 | 54.64 | 46.18 |
| PyramidKV | 30.33 | 38.03 | 55.11 | 57.17 | 48.65 | 31.26 | 25.28 | 24.73 | 19.79 | 9.25 | 100 | 43 | 91.68 | 42.04 | 61.98 | 54.61 | 45.81 |
| StreamingLLM | 27.82 | 24.8 | 33.13 | 50.57 | 42.3 | 24.17 | 24.06 | 21.69 | 19.06 | 9.5 | 92.5 | 42.5 | 87 | 41.32 | 61.45 | 51.98 | 40.87 |
| KeyDiff | 30.41 | 33.2 | 44.1 | 53.59 | 41.45 | 28.98 | 25.33 | 23.22 | 20.94 | 9.5 | 74.5 | 44 | 88.38 | 30.44 | 36.82 | 35.82 | 38.79 |
| CAKE | 29.6 | 40.32 | 56.08 | 57.69 | 49.45 | 31.18 | 26.11 | 24.56 | 20.03 | 9.5 | 100 | 50 | 92.53 | 43.72 | 62.05 | 54.98 | 46.74 |
| CriticalKV | 30.39 | 38.44 | 55.87 | 57.22 | 48.46 | 31.85 | 26.29 | 24.9 | 20.96 | 9.25 | 100 | 55 | 91.6 | 42.94 | 63.15 | 54.42 | 46.92 |
| AnDPro | 30.9 | 37.39 | 56.18 | 57.9 | 48.13 | 32.07 | 25.46 | 24.38 | 20.48 | 9.25 | 100 | 56 | 91.97 | 43.77 | 61.74 | 55.64 | 46.95 |
| DropKV | 31.34 | 40.66 | 57.3 | 57.9 | 48.49 | 31.65 | 26.77 | 24.37 | 21 | 9.5 | 100 | 59 | 91.75 | 43.61 | 62.29 | 54.38 | **47.50** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | | | | | |
| FullKV | 28.88 | 48.39 | 51.57 | 60.27 | 53.19 | 35.63 | 33.33 | 23.22 | 24.53 | 6.5 | 100 | 66.5 | 85.28 | 45.29 | 45.61 | 35.31 | 46.47 |
| SnapKV | 28.02 | 42.45 | 50.51 | 60.51 | 52.39 | 35.34 | 26.94 | 22.9 | 16.65 | 6.5 | 100 | 44.67 | 85.91 | 44.39 | 42.28 | 35.6 | 43.44 |
| PyramidKV | 27.25 | 40.68 | 49.42 | 57.89 | 51.61 | 34.66 | 24.32 | 21.95 | 16.57 | 7 | 100 | 35.67 | 84.73 | 44.38 | 41.51 | 32.58 | 41.89 |
| StreamingLLM | 23.09 | 32.37 | 28.91 | 46.45 | 45.03 | 24.28 | 24.61 | 19.19 | 16.52 | 6 | 24.5 | 38.5 | 82.74 | 43.18 | 39.9 | 34.78 | 33.13 |
| KeyDiff | 24.73 | 27.54 | 31.99 | 44.68 | 34 | 23.41 | 23.55 | 20.23 | 14.54 | 6 | 53.5 | 32.54 | 67.15 | 23.45 | 23.63 | 20.36 | 29.46 |
| CAKE | 28.85 | 41.88 | 50.94 | 60.09 | 52.09 | 34.75 | 27.41 | 22.97 | 17.23 | 7 | 100 | 48 | 85.03 | 44.31 | 43.09 | 35.35 | 43.69 |
| CriticalKV | 27.93 | 42.81 | 50.71 | 60.55 | 52.34 | 35.1 | 27.67 | 22.69 | 17.52 | 7 | 100 | 48.5 | 85.39 | 44.22 | 42.46 | 35.64 | 43.78 |
| AnDPro | 27.98 | 42.07 | 50.68 | 59.84 | 51.75 | 36.02 | 26.97 | 22.86 | 16.56 | 7 | 100 | 53 | 85.34 | 45.1 | 42.65 | 35.23 | 43.94 |
| DropKV | 28.48 | 43.43 | 51 | 60.08 | 52.84 | 34.46 | 28.23 | 22.15 | 18.14 | 6.5 | 100 | 59.5 | 85.15 | 44.62 | 43.01 | 35.73 | **44.58** |
| **Mistral-7B-512k** | | | | | | | | | | | | | | | | | |
| FullKV | 22.04 | 16.47 | 42.75 | 24.06 | 20.72 | 13.04 | 33.35 | 25.25 | 26.97 | 0 | 100 | 74.5 | 89.55 | 41.69 | 55.54 | 56.39 | 40.15 |
| SnapKV | 21.18 | 12.48 | 40.73 | 19.75 | 16.58 | 10.24 | 26.92 | 24.97 | 21.49 | 0 | 100 | 58.5 | 90.3 | 40.4 | 53.55 | 54.6 | 36.98 |
| PyramidKV | 21.44 | 13.73 | 40.92 | 17.38 | 16.41 | 9.38 | 25.8 | 24.77 | 21.44 | 0 | 100 | 55 | 90.3 | 40.71 | 53.29 | 53.75 | 36.52 |
| StreamingLLM | 19.85 | 9.35 | 26.55 | 14.24 | 15.21 | 6.83 | 22.28 | 22.11 | 19.18 | 0.5 | 32 | 47 | 88.72 | 39.82 | 51.23 | 51.66 | 29.16 |
| KeyDiff | 20.64 | 12.21 | 26.77 | 16.57 | 14 | 7.14 | 24.66 | 23.21 | 21.78 | 0 | 16 | 26.5 | 74.6 | 28.22 | 29.6 | 30.2 | 23.26 |
| CAKE | 21.83 | 13.39 | 40.09 | 20.51 | 17.09 | 11.52 | 27.51 | 25.21 | 22.11 | 0 | 100 | 60.5 | 90.13 | 40.38 | 52.47 | 55.35 | 37.38 |
| CriticalKV | 22.9 | 12.9 | 40.14 | 20.21 | 16.17 | 10.7 | 27.58 | 25.18 | 22.47 | 0 | 100 | 60 | 90.3 | 40.08 | 53.19 | 54.75 | 37.29 |
| AnDPro | 21.37 | 12.46 | 41.25 | 20.06 | 16.26 | 9.84 | 26.81 | 25.07 | 21.37 | 0 | 100 | 63.5 | 90.3 | 40.57 | 53.26 | 54.24 | 37.27 |
| DropKV | 21.62 | 13.49 | 41.16 | 21.75 | 16.02 | 10.5 | 28.11 | 25.11 | 22.85 | 0 | 100 | 69 | 90.3 | 40.55 | 53.44 | 54.11 | **38.00** |

*Table 4.* Additional LongBench results at 30% budget. The best average result is highlighted in gray, and the second-best is underlined.

| | Single Doc. QA | | | Multi Doc. QA | | | Summarization | | | Synthetic | | Fewshot Learning | | | Code | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | NrtvQA | Qasper | MF-en | HotpotQA | 2WikiMQA | Musique | GovReport | QMSum | MultiNews | PCount | PR-en | TREC | TriviaQA | SAMSum | Lcc | RB-P | Avg. |
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | | | | | |
| FullKV | 30.51 | 45.59 | 56.34 | 58.22 | 50.02 | 31.75 | 34.3 | 25.33 | 26.76 | 9.5 | 100 | 73.5 | 92.29 | 43.96 | 64.77 | 55.7 | 49.91 |
| SnapKV | 30.28 | 45.93 | 56.13 | 58.05 | 50.38 | 32.07 | 30.58 | 24.96 | 23.74 | 9.5 | 100 | 65.5 | 91.87 | 42.71 | 64.83 | 55.95 | 48.91 |
| PyramidKV | 30.58 | 44.36 | 55.8 | 57.98 | 49.83 | 31.7 | 30.06 | 24.99 | 23.53 | 9.5 | 100 | 62.5 | 91.88 | 43.42 | 64.37 | 55.68 | 48.51 |
| StreamingLLM | 28.33 | 31.55 | 37.18 | 54.34 | 48.13 | 26.98 | 28.64 | 22.44 | 23.25 | 9.5 | 83.5 | 59.5 | 91.54 | 43.42 | 63.18 | 54.13 | 44.10 |
| KeyDiff | 30.23 | 40.88 | 54.35 | 57.59 | 48.15 | 30.29 | 29.87 | 24.58 | 24.14 | 9.17 | 99.5 | 61.5 | 90.74 | 41.1 | 49.15 | 48.17 | 46.21 |
| CAKE | 29.65 | 44.72 | 56.87 | 57.98 | 49.5 | 32.1 | 30.29 | 24.99 | 23.69 | 9.5 | 100 | 66.5 | 92.26 | 43.44 | 64.4 | 55.21 | 48.82 |
| CriticalKV | 30.61 | 45.54 | 56.57 | 57.93 | 49.89 | 32.04 | 30.92 | 24.95 | 24.12 | 9.5 | 100 | 68.5 | 91.87 | 42.83 | 64.63 | 55.12 | 49.06 |
| AnDPro | 30.78 | 44.93 | 56.67 | 57.95 | 49.58 | 31.95 | 30.56 | 25.39 | 23.72 | 9.5 | 100 | 67.5 | 92.33 | 43.51 | 64.62 | 55.95 | 49.06 |
| DropKV | 30.29 | 45.22 | 57.22 | 58.02 | 50.06 | 31.42 | 31.07 | 24.93 | 24.13 | 9.5 | 100 | 68.5 | 91.73 | 44.18 | 64.46 | 55.57 | **49.14** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | | | | | |
| FullKV | 28.88 | 48.39 | 51.57 | 60.27 | 53.19 | 35.63 | 33.33 | 23.22 | 24.53 | 6.5 | 100 | 66.5 | 85.28 | 45.29 | 45.61 | 35.31 | 46.47 |
| SnapKV | 28.44 | 46.63 | 51.16 | 59.98 | 53.11 | 36.02 | 31.17 | 23.35 | 20.53 | 7.5 | 100 | 61.5 | 85.03 | 45.04 | 45.77 | 35.74 | 45.69 |
| PyramidKV | 29.12 | 42.89 | 51.19 | 59.8 | 52.79 | 35.42 | 27.35 | 22.43 | 19.05 | 7 | 100 | 48 | 84.67 | 44.74 | 43.81 | 34.52 | 43.92 |
| StreamingLLM | 25.91 | 37.66 | 32.18 | 50.75 | 48.04 | 27.35 | 29.46 | 19.84 | 20.97 | 5.5 | 46 | 54.5 | 85.29 | 44.41 | 44.31 | 36.62 | 38.05 |
| KeyDiff | 27.13 | 37.4 | 44.17 | 55.13 | 46.07 | 30.67 | 29.02 | 21.94 | 19.79 | 5 | 100 | 61 | 82.8 | 40.72 | 31.11 | 25.05 | 41.06 |
| CAKE | 28.81 | 46.36 | 51.38 | 59.78 | 53.34 | 35.62 | 31.22 | 23.18 | 20.95 | 7.5 | 100 | 63 | 84.85 | 45.48 | 45.74 | 35.49 | 45.79 |
| CriticalKV | 28.52 | 46.75 | 51.72 | 59.95 | 52.9 | 35.65 | 31.37 | 22.84 | 20.94 | 6.5 | 100 | 64.5 | 84.99 | 45.26 | 45.32 | 35.9 | 45.82 |
| AnDPro | 27.77 | 46.58 | 50.75 | 60.3 | 53.41 | 35.55 | 31.37 | 23.14 | 20.6 | 7 | 100 | 66 | 84.99 | 44.92 | 45.4 | 35.84 | 45.85 |
| DropKV | 28.28 | 47.4 | 51.18 | 60.13 | 52.84 | 35.12 | 31.55 | 22.86 | 21.55 | 7 | 100 | 65 | 84.99 | 45.13 | 45.52 | 35.99 | **45.91** |
| **Mistral-7B-512k** | | | | | | | | | | | | | | | | | |
| FullKV | 22.04 | 16.47 | 42.75 | 24.06 | 20.72 | 13.04 | 33.35 | 25.25 | 26.97 | 0 | 100 | 74.5 | 89.55 | 41.69 | 55.54 | 56.39 | 40.15 |
| SnapKV | 21.88 | 16.3 | 42.1 | 21.77 | 19.02 | 11.4 | 30.74 | 25.33 | 24.5 | 0 | 100 | 69 | 89.71 | 40.78 | 54.93 | 56.14 | 38.98 |
| PyramidKV | 21.92 | 17.28 | 41.48 | 18.51 | 15.3 | 10.68 | 30.05 | 25.63 | 24.08 | 0 | 100 | 66.5 | 90.05 | 41.1 | 54 | 54.5 | 38.19 |
| StreamingLLM | 21.53 | 12.1 | 27.22 | 18.28 | 16.83 | 8.58 | 27.68 | 22.44 | 23.4 | 0 | 63 | 60.5 | 89.38 | 39.97 | 53.96 | 54.68 | 33.72 |
| KeyDiff | 22.82 | 15.87 | 36.11 | 18.54 | 15.44 | 9.29 | 28.75 | 24.98 | 24.81 | 0 | 71.5 | 56 | 85.25 | 38.3 | 41.82 | 47.91 | 33.59 |
| CAKE | 21.46 | 15.56 | 41.85 | 20.99 | 19.32 | 11.69 | 31.28 | 25.05 | 24.91 | 0 | 100 | 68.5 | 89.71 | 41.83 | 54.56 | 56.51 | 38.95 |
| CriticalKV | 22.22 | 15.62 | 41.47 | 21.9 | 19.95 | 11.38 | 30.98 | 25.41 | 25.02 | 0 | 100 | 72.5 | 89.96 | 41.4 | 55.36 | 56.01 | 39.32 |
| AnDPro | 21.18 | 15.97 | 41.65 | 22.16 | 18.72 | 11.67 | 30.59 | 25.16 | 24.84 | 0 | 100 | 73.5 | 89.79 | 42.28 | 54.94 | 56.1 | 39.28 |
| DropKV | 22.71 | 15.8 | 41.1 | 22.34 | 19.82 | 11.82 | 30.98 | 25.5 | 25.28 | 0 | 100 | 73.5 | 89.71 | 42.48 | 55.08 | 55.78 | **39.49** |

Tables 5–8 report all RULER evaluation across different sequence lengths ($n \in \{16\text{K}, 32\text{K}, 64\text{K}\,128\text{K}\}$ at 5%, 10%, and 30% budgets).

*Table 5.* RULER results at 128K sequence length 5%, 10% and 30% KV-cache budgets. The best result is highlighted in gray, and the second-best is underlined.

| **Method** | Budget | N-S1 | N-S2 | N-S3 | N-MK1 | N-MK2 | N-MQ | N-MV | VT | FWE | QA-1 | QA-2 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 99.2 | 99.6 | 97.6 | 88.2 | 98.7 | 95.85 | 84 | 61.93 | 64.17 | 44 | 84.84 |
| SnapKV | 5% | 100 | 99 | 71.4 | 97.6 | 76.2 | 98.6 | 95.75 | 76.92 | 52.53 | 64.03 | 43.2 | 79.57 |
| PyramidKV | | 100 | 99.4 | 62.4 | 97.4 | 74.2 | 98.6 | 94.7 | 78.36 | 54.86 | 63.9 | 45.6 | 79.04 |
| StreamingLLM | | 4.4 | 4.2 | 4.8 | 5.6 | 5 | 5.05 | 5 | 4.9 | 73.4 | 37.48 | 37.4 | 17.02 |
| KeyDiff | | 100 | 98.4 | 16.6 | 81.6 | 0 | 78.3 | 75.55 | 80.2 | 26.6 | 42.58 | 36.8 | 57.88 |
| CAKE | | 100 | 99 | 78.4 | 97.4 | 74.2 | 98.05 | 92.95 | 75.28 | 44.73 | 63.63 | 43.2 | 78.80 |
| CriticalKV | | 100 | 99 | 69.2 | 97.6 | 75.6 | 98.55 | 95.2 | 77.4 | 46.67 | 63.9 | 42.8 | 78.72 |
| AnDPro | | 100 | 99 | 68.4 | 97.4 | 78.4 | 98.75 | 96 | 76.52 | 54.53 | 63.76 | 43.4 | 79.65 |
| DropKV | | 100 | 98.8 | 89.6 | 97.6 | 86.2 | 98.25 | 96.9 | 77.32 | 49.73 | 64.17 | 43.4 | **82.00** |
| SnapKV | 10% | 100 | 99 | 90.8 | 97.4 | 78.2 | 98.45 | 96.4 | 78.92 | 56.47 | 63.23 | 43.8 | 82.06 |
| PyramidKV | | 100 | 100 | 55.47 | 97.66 | 71.88 | 98.83 | 93.75 | 76.09 | 53.91 | 60.81 | 42.97 | 77.40 |
| StreamingLLM | | 11.72 | 8.59 | 14.84 | 11.72 | 12.50 | 10.55 | 8.79 | 10.94 | 75.78 | 34.83 | 32.82 | 21.19 |
| KeyDiff | | 100 | 100 | 57.03 | 94.53 | 1.56 | 95.51 | 94.92 | 78.44 | 45.31 | 43.68 | 35.16 | 67.83 |
| CAKE | | 100 | 99 | 94.2 | 97.6 | 80 | 98.7 | 95.35 | 77.08 | 54.4 | 63.57 | 43.6 | 82.14 |
| CriticalKV | | 100 | 98.8 | 90.4 | 97.4 | 81.8 | 98.5 | 95.55 | 78.64 | 52.73 | 63.3 | 43.4 | 81.87 |
| AnDPro | | 100 | 99 | 89.2 | 97.4 | 82.6 | 99 | 95.6 | 78.96 | 56.27 | 63.3 | 43.8 | 82.28 |
| DropKV | | 100 | 98.6 | 97.6 | 97.6 | 87.6 | 98.8 | 97.1 | 79.24 | 51.87 | 64.16 | 43.4 | **83.27** |
| SnapKV | 30% | 100 | 100 | 99.22 | 97.66 | 84.38 | 99.22 | 94.09 | 78.44 | 57.9 | 59.24 | 41.41 | 82.87 |
| PyramidKV | | 100 | 100 | 99.22 | 97.66 | 85.94 | 99.22 | 94.34 | 79.22 | 56.25 | 59.51 | 42.19 | 83.05 |
| StreamingLLM | | 32.03 | 28.13 | 34.38 | 35.16 | 26.56 | 31.84 | 29.1 | 32.81 | 80.73 | 67.32 | 36.72 | 39.53 |
| KeyDiff | | 100 | 100 | 99.22 | 98.44 | 17.19 | 99.22 | 100 | 77.97 | 58.59 | 50 | 37.5 | 76.19 |
| CAKE | | 100 | 100 | 99.22 | 97.66 | 85.16 | 99.22 | 94.92 | 78.44 | 60.24 | 59.24 | 41.41 | 83.23 |
| CriticalKV | | 100 | 100 | 99.22 | 97.66 | 85.94 | 99.22 | 95.51 | 77.26 | 60.68 | 59.25 | 41.41 | 83.29 |
| AnDPro | | 100 | 100 | 97.66 | 97.66 | 84.38 | 99.02 | 93.55 | 76.56 | 58.33 | 60.55 | 41.41 | 82.65 |
| DropKV | | 100 | 100 | 99.22 | 97.66 | 85.94 | 99.22 | 96.09 | 77.5 | 59.38 | 60.03 | 41.41 | **83.31** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 99.6 | 99.8 | 98.2 | 99.9 | 78.9 | 95.6 | 81.13 | 57 | 47 | 87.01 |
| SnapKV | 5% | 100 | 100 | 16.4 | 99.6 | 86.2 | 99.55 | 77.1 | 96.2 | 66.4 | 56.05 | 46 | 76.68 |
| PyramidKV | | 100 | 100 | 4 | 99.6 | 78.6 | 99.55 | 77.25 | 96.16 | 62.87 | 55.78 | 44.4 | 74.38 |
| StreamingLLM | | 4.8 | 4.6 | 4.8 | 5.2 | 4.8 | 4.65 | 4.8 | 7.35 | 81.6 | 28.1 | 35.6 | 16.94 |
| KeyDiff | | 100 | 98.2 | 63.8 | 64 | 0.4 | 53.8 | 51.7 | 94.64 | 55.93 | 25.88 | 22.6 | 57.36 |
| CAKE | | 100 | 100 | 25.6 | 99.6 | 25.8 | 99.6 | 76.45 | 96.2 | 65.67 | 56.47 | 46.6 | 72.00 |
| CriticalKV | | 100 | 100 | 22.6 | 99.8 | 88.2 | 99.55 | 77.15 | 96.48 | 64.93 | 56.38 | 46.6 | 77.43 |
| AnDPro | | 99.8 | 99.8 | 45.8 | 99.6 | 90 | 99.6 | 77.25 | 96.4 | 66.93 | 57.73 | 46 | 79.90 |
| DropKV | | 100 | 99.6 | 81.6 | 99.2 | 97.4 | 99.7 | 77.15 | 96.64 | 67.33 | 57.07 | 45.8 | **83.77** |
| SnapKV | 10% | 100 | 100 | 42.6 | 99.8 | 87.8 | 99.6 | 76.55 | 96.32 | 70.53 | 56.67 | 46.6 | 79.68 |
| PyramidKV | | 100 | 100 | 11 | 99.8 | 82.8 | 99.7 | 76.95 | 96.24 | 66.13 | 55.58 | 45 | 75.75 |
| StreamingLLM | | 7.81 | 8.59 | 12.50 | 8.59 | 13.28 | 9.96 | 8.98 | 15.16 | 84.11 | 23.50 | 37.50 | 20.91 |
| KeyDiff | | 100 | 99.22 | 92.97 | 85.16 | 2.34 | 81.84 | 73.24 | 95.47 | 58.85 | 25.59 | 31.25 | 67.81 |
| CAKE | | 100 | 100 | 60.4 | 99.6 | 37.2 | 99.75 | 76.95 | 95.88 | 69.8 | 56.2 | 45.8 | 76.51 |
| CriticalKV | | 100 | 100 | 58.2 | 99.6 | 92 | 99.6 | 76.65 | 96.12 | 69.13 | 56.73 | 45.8 | 81.26 |
| AnDPro | | 99.8 | 100 | 74 | 99.8 | 94.6 | 99.75 | 76.9 | 96.48 | 71.4 | 57.4 | 45.6 | 83.25 |
| DropKV | | 100 | 99.4 | 97 | 99.2 | 97.8 | 99.7 | 77.1 | 96.48 | 69.6 | 56.13 | 46 | **85.31** |
| SnapKV | 30% | 100 | 100 | 92.97 | 100 | 96.09 | 99.81 | 75.98 | 96.09 | 76.82 | 56.38 | 47.66 | 85.62 |
| PyramidKV | | 100 | 100 | 50 | 100 | 89.84 | 100 | 76.56 | 95.31 | 72.14 | 56.38 | 47.66 | 80.72 |
| StreamingLLM | | 30.47 | 32.81 | 35.16 | 30.47 | 32.03 | 30.66 | 28.52 | 47.66 | 84.9 | 59.44 | 38.28 | 40.95 |
| KeyDiff | | 100 | 100 | 100 | 98.44 | 16.41 | 100 | 81.45 | 95.16 | 67.97 | 35.48 | 36.72 | 75.60 |
| CAKE | | 100 | 100 | 99.22 | 100 | 79.69 | 99.81 | 75.98 | 95.94 | 77.34 | 56.64 | 45.31 | 84.54 |
| CriticalKV | | 100 | 100 | 98.44 | 100 | 97.66 | 99.8 | 75.39 | 96.41 | 74.74 | 54.82 | 47.66 | 85.90 |
| AnDPro | | 100 | 100 | 89.06 | 100 | 97.66 | 99.61 | 75.97 | 95.78 | 78.39 | 55.6 | 48.44 | 85.50 |
| DropKV | | 100 | 100 | 99.22 | 100 | 99.22 | 99.61 | 75.78 | 96.09 | 73.96 | 57.16 | 47.66 | **86.25** |
| **Mistral-7B-512K** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 99.6 | 99.2 | 75.4 | 96.48 | 90.8 | 68.78 | 49 | 89.02 |
| SnapKV | 5% | 100 | 95.6 | 42.2 | 98 | 47.8 | 97.5 | 77.3 | 93.64 | 87.33 | 68.18 | 47.4 | 77.72 |
| PyramidKV | | 100 | 95.8 | 36 | 98.4 | 46.2 | 98.1 | 78.6 | 94.56 | 87.93 | 68.25 | 47.4 | 77.39 |
| StreamingLLM | | 5.6 | 6.4 | 7.8 | 7.2 | 6 | 6.05 | 6.2 | 7.68 | 91.2 | 33.23 | 32.2 | 19.05 |
| KeyDiff | | 100 | 64.8 | 20 | 46.6 | 1.2 | 27.35 | 31.95 | 93.36 | 73 | 34.8 | 19.6 | 46.61 |
| CAKE | | 100 | 97.2 | 56.8 | 97.6 | 29 | 97.85 | 78 | 93.24 | 84.93 | 68.12 | 47.8 | 77.32 |
| CriticalKV | | 100 | 98 | 42.2 | 99.4 | 47.4 | 97.55 | 78.05 | 93.48 | 86.87 | 68.32 | 47.4 | 78.06 |
| AnDPro | | 100 | 98.4 | 56.2 | 99.2 | 62.2 | 98.35 | 80.2 | 94.12 | 90.06 | 68.97 | 47.4 | 81.37 |
| DropKV | | 100 | 96.4 | 76.4 | 99 | 92.2 | 97.05 | 79.95 | 93.48 | 88.93 | 68.63 | 47.4 | **85.40** |
| SnapKV | 10% | 100 | 96 | 72.8 | 98.6 | 38.8 | 98.5 | 79.7 | 94.8 | 86.27 | 68.12 | 47 | 80.05 |
| PyramidKV | | 100 | 97.4 | 66.4 | 99.2 | 40.2 | 98.9 | 78.7 | 95.16 | 87.53 | 68.12 | 47.4 | 79.91 |
| StreamingLLM | | 7.03 | 14.84 | 14.06 | 11.72 | 10.16 | 11.13 | 12.31 | 12.50 | 90.89 | 28.71 | 28.91 | 22.02 |
| KeyDiff | | 100 | 89.84 | 64.84 | 75.78 | 3.91 | 73.44 | 57.42 | 94.06 | 78.13 | 41.02 | 24.22 | 63.88 |
| CAKE | | 100 | 97.2 | 84.2 | 98.8 | 29 | 98.85 | 78.5 | 93.88 | 83.8 | 67.78 | 47.6 | 79.96 |
| CriticalKV | | 100 | 98 | 73 | 99.2 | 41.4 | 98.45 | 78.55 | 94.16 | 87.4 | 67.73 | 47 | 80.44 |
| AnDPro | | 100 | 98 | 85.4 | 99.6 | 48.4 | 99.1 | 79.95 | 94.44 | 90.47 | 67.78 | 47.8 | 82.81 |
| DropKV | | 100 | 97 | 88.4 | 100 | 93.2 | 98.2 | 79.05 | 94.24 | 88.2 | 67.77 | 47.8 | **86.71** |
| SnapKV | 30% | 100 | 100 | 98.44 | 100 | 42.19 | 99.41 | 75.98 | 95.62 | 87.24 | 71.74 | 49.22 | 83.62 |
| PyramidKV | | 100 | 100 | 96.88 | 100 | 55.47 | 99.61 | 75.78 | 95.78 | 86.46 | 70.18 | 48.44 | 84.42 |
| StreamingLLM | | 30.47 | 27.34 | 31.25 | 26.56 | 31.25 | 27.34 | 31.05 | 40.78 | 90.63 | 73.83 | 36.72 | 40.66 |
| KeyDiff | | 100 | 98.44 | 98.44 | 94.53 | 21.88 | 98.24 | 72.85 | 95.62 | 80.99 | 56.12 | 37.5 | 77.69 |
| CAKE | | 100 | 99.22 | 100 | 100 | 36.72 | 99.61 | 76.17 | 95.62 | 85.16 | 71.22 | 49.22 | 82.99 |
| CriticalKV | | 100 | 100 | 100 | 100 | 56.25 | 99.41 | 75.78 | 94.53 | 88.8 | 70.96 | 48.44 | 84.92 |
| AnDPro | | 100 | 99.22 | 99.22 | 100 | 48.44 | 99.41 | 76.95 | 95.62 | 90.89 | 70.96 | 49.22 | 84.54 |
| DropKV | | 100 | 97.66 | 100 | 99.22 | 97.65 | 99.02 | 78.32 | 95.62 | 87.76 | 70.96 | 49.22 | **88.68** |

*Table 6.* RULER results at 64K sequence length and 5%, 10% and 30% KV-cache budget. The best result is highlighted in gray, and the second-best is underlined.

| **Method** | Budget | N-S1 | N-S2 | N-S3 | N-MK1 | N-MK2 | N-MQ | N-MV | VT | FWE | QA-1 | QA-2 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | |
| FullKV | 100% | 100 | 100 | 100 | 100 | 100 | 100 | 99.02 | 98.91 | 85.68 | 71.22 | 50 | 91.35 |
| SnapKV | 5% | 100 | 99.40 | 84.6 | 99 | 94.4 | 99.5 | 98.7 | 96.24 | 74.87 | 69.9 | 49.2 | 87.80 |
| PyramidKV | | 100 | 100.00 | 57.8 | 99.6 | 94.8 | 99.5 | 99.1 | 97.24 | 73.13 | 69.9 | 49.8 | 85.53 |
| StreamingLLM | | 5.80 | 7.40 | 4.80 | 7.60 | 5.40 | 4.25 | 5.05 | 9.12 | 94.87 | 35.80 | 40.77 | 20.08 |
| KeyDiff | | 100 | 99.40 | 42.40 | 89.80 | 1.40 | 87.35 | 90.55 | 95.20 | 54.53 | 38 | 45.85 | 67.68 |
| CAKE | | 100 | 99.00 | 82.4 | 98.6 | 90.6 | 98.45 | 96.4 | 96.4 | 71.27 | 68.57 | 49 | 86.43 |
| CriticalKV | | 100 | 100.00 | 76.2 | 99.2 | 95.4 | 99.55 | 99.15 | 96.24 | 72.73 | 69.3 | 49.2 | 87.00 |
| AnDPro | | 100 | 100.00 | 76.8 | 99.8 | 97.6 | 99.85 | 98.7 | 96.92 | 74.53 | 69.67 | 49.4 | 87.57 |
| DropKV | | 100 | 100.00 | 96.8 | 99.6 | 99 | 99.45 | 98.35 | 96.2 | 71.67 | 69.2 | 50.4 | **89.15** |
| SnapKV | 10% | 100 | 100 | 99.22 | 99.22 | 98.44 | 99.61 | 99.22 | 97.03 | 80.02 | 70.44 | 46.09 | 89.94 |
| PyramidKV | | 100 | 100 | 91.41 | 100 | 96.88 | 99.61 | 99.81 | 97.5 | 78.91 | 72.79 | 50.78 | 89.79 |
| StreamingLLM | | 13.28 | 12.5 | 7.81 | 10.16 | 17.19 | 8.98 | 10.55 | 15.16 | 96.35 | 44.08 | 37.5 | 24.87 |
| KeyDiff | | 100 | 100 | 95.31 | 97.66 | 3.91 | 99.22 | 98.63 | 95.47 | 64.58 | 50.59 | 43.75 | 77.19 |
| CAKE | | 100 | 99.22 | 99.22 | 99.22 | 96.09 | 99.81 | 98.05 | 97.03 | 77.6 | 70.96 | 49.22 | 89.67 |
| CriticalKV | | 100 | 100 | 96.09 | 100 | 99.22 | 99.81 | 99.61 | 96.88 | 80.21 | 70.44 | 47.66 | 89.99 |
| AnDPro | | 100 | 100 | 96.09 | 100 | 98.44 | 100 | 99.41 | 97.18 | 78.85 | 71.74 | 48.44 | 90.01 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 100 | 99.22 | 97.19 | 77.87 | 72.01 | 46.88 | **90.29** |
| SnapKV | 30% | 100 | 100 | 100 | 100 | 100 | 100 | 99.41 | 98.13 | 85.94 | 70.44 | 48.44 | 91.12 |
| PyramidKV | | 100 | 100 | 100 | 100 | 99.22 | 100 | 98.83 | 98.44 | 84.38 | 71.48 | 48.44 | 90.98 |
| StreamingLLM | | 34.38 | 35.17 | 31.25 | 33.59 | 34.38 | 30.47 | 31.45 | 36.72 | 95.05 | 39.13 | 38.28 | 39.99 |
| KeyDiff | | 100 | 100 | 100 | 100 | 29.69 | 100 | 99.61 | 97.34 | 86.2 | 53.52 | 43.75 | 82.74 |
| CAKE | | 100 | 100 | 99.22 | 100 | 99.22 | 99.81 | 99.61 | 98.12 | 86.2 | 70.44 | 47.66 | 90.93 |
| CriticalKV | | 100 | 100 | 100 | 100 | 100 | 100 | 98.63 | 97.97 | 85.42 | 71.22 | 46.88 | 90.92 |
| AnDPro | | 100 | 100 | 100 | 100 | 100 | 100 | 99.81 | 98.12 | 86.58 | 70.7 | 47.66 | 91.17 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 100 | 99.02 | 98.28 | 86.46 | 71.75 | 48.09 | **91.24** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | |
| FullKV | 100% | 100 | 100 | 100 | 100 | 99.22 | 100 | 80.86 | 94.53 | 83.33 | 60.29 | 57.81 | 88.73 |
| SnapKV | 5% | 100 | 100 | 10.2 | 100 | 83.4 | 99.9 | 77.8 | 94.84 | 66.87 | 55.38 | 52.2 | 76.42 |
| PyramidKV | | 100 | 100 | 1 | 99.8 | 73 | 99.8 | 77.5 | 92.4 | 66.67 | 55.57 | 51.6 | 74.30 |
| StreamingLLM | | 5.80 | 5.60 | 4.40 | 3.80 | 4.80 | 4.85 | 4.75 | 9.32 | 81.13 | 32.40 | 29.35 | 16.93 |
| KeyDiff | | 100 | 99.80 | 91 | 78.20 | 0.80 | 65.45 | 63.45 | 92.68 | 66.13 | 23.40 | 21.75 | 63.88 |
| CAKE | | 100 | 100 | 14 | 99.6 | 19 | 99.45 | 75.7 | 94.72 | 67.27 | 55 | 52 | 70.61 |
| CriticalKV | | 100 | 100 | 12.6 | 100 | 88.2 | 99.9 | 77.55 | 94.84 | 66.73 | 55.38 | 52 | 77.02 |
| AnDPro | | 100 | 99.8 | 39.2 | 100 | 89.4 | 99.8 | 78.7 | 95.36 | 67 | 55.33 | 52.4 | 79.73 |
| DropKV | | 100 | 99.6 | 74.4 | 99.6 | 98.4 | 99.85 | 78.95 | 95.36 | 67.13 | 55.73 | 51.8 | **83.71** |
| SnapKV | 10% | 100 | 100 | 28.13 | 100 | 88.28 | 100 | 79.1 | 93.75 | 67.19 | 58.2 | 57.03 | 79.24 |
| PyramidKV | | 100 | 100 | 3.91 | 100 | 80.47 | 100 | 79.1 | 94.06 | 66.67 | 58.14 | 56.25 | 76.24 |
| StreamingLLM | | 9.38 | 7.81 | 5.47 | 7.03 | 9.38 | 8.4 | 10.55 | 19.37 | 83.07 | 26.89 | 35.16 | 20.23 |
| KeyDiff | | 100 | 100 | 100 | 93.75 | 1.56 | 91.8 | 77.73 | 95.47 | 66.67 | 23.76 | 31.25 | 71.09 |
| CAKE | | 100 | 100 | 48.44 | 100 | 32.81 | 100 | 77.92 | 94.69 | 67.19 | 57.68 | 56.25 | 75.91 |
| CriticalKV | | 100 | 100 | 42.97 | 100 | 91.41 | 100 | 77.93 | 94.06 | 67.19 | 56.84 | 58.59 | 80.82 |
| AnDPro | | 100 | 99.22 | 64.84 | 100 | 94.53 | 99.81 | 78.52 | 95.16 | 67.19 | 57.42 | 56.25 | 82.99 |
| DropKV | | 100 | 99.22 | 85.94 | 100 | 98.44 | 100 | 79.1 | 94.69 | 68.49 | 58.72 | 56.25 | **85.53** |
| SnapKV | 30% | 100 | 99.22 | 83.59 | 100 | 96.09 | 100 | 78.32 | 94.06 | 72.14 | 58.2 | 57.03 | 85.33 |
| PyramidKV | | 100 | 100 | 39.84 | 100 | 92.97 | 99.81 | 80.27 | 94.69 | 69.79 | 58.46 | 55.47 | 81.03 |
| StreamingLLM | | 29.69 | 25 | 24.22 | 30.47 | 30.47 | 25.78 | 34.77 | 47.03 | 85.68 | 26.3 | 35.16 | 35.87 |
| KeyDiff | | 100 | 99.22 | 100 | 100 | 16.41 | 100 | 82.62 | 94.84 | 69.53 | 34.96 | 46.09 | 76.70 |
| CAKE | | 100 | 100 | 95.31 | 100 | 70.31 | 99.61 | 79.69 | 93.75 | 72.14 | 57.68 | 57.81 | 84.21 |
| CriticalKV | | 100 | 100 | 96.09 | 100 | 97.66 | 99.81 | 78.91 | 94.69 | 74.48 | 57.88 | 57.03 | 86.96 |
| AnDPro | | 100 | 100 | 91.41 | 100 | 96.09 | 99.61 | 79.69 | 94.53 | 73.44 | 58.2 | 56.25 | 86.29 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 99.41 | 79.1 | 94.69 | 75.26 | 58.72 | 57.03 | **87.66** |
| **Mistral-7B-512K** | | | | | | | | | | | | | |
| FullKV | 100% | 100 | 100 | 100 | 99.22 | 99.22 | 99.81 | 75 | 96.41 | 79.43 | 70.44 | 53.91 | 88.49 |
| SnapKV | 5% | 100 | 99.4 | 48 | 99.2 | 67.2 | 98.45 | 75.1 | 96.56 | 74.67 | 70.97 | 50.8 | 80.03 |
| PyramidKV | | 100 | 99.6 | 30.4 | 99.6 | 66.4 | 98.65 | 76.85 | 96.2 | 73.53 | 70.83 | 50.6 | 78.42 |
| StreamingLLM | | 6.60 | 4.40 | 6.80 | 7.20 | 5.80 | 8.75 | 6.15 | 7.60 | 92.60 | 29.80 | 35.83 | 19.23 |
| KeyDiff | | 100 | 78.80 | 23.40 | 58.80 | 0.60 | 34.35 | 35.10 | 95.76 | 69.93 | 22.40 | 33.20 | 50.21 |
| CAKE | | 100 | 99.6 | 66.2 | 99 | 50.8 | 99 | 75.95 | 95.32 | 74.87 | 70.17 | 50.4 | 80.12 |
| CriticalKV | | 100 | 99.6 | 44 | 98.8 | 65.6 | 98.45 | 75.9 | 95.6 | 73.67 | 70.63 | 51.4 | 79.42 |
| AnDPro | | 100 | 99.4 | 64.2 | 99.8 | 82.4 | 99.2 | 78.15 | 96.16 | 74.73 | 71.23 | 53.2 | 83.50 |
| DropKV | | 100 | 99.2 | 69.2 | 99.4 | 93.8 | 97.75 | 77.4 | 96.28 | 73.93 | 69.77 | 51.4 | 84.38 |
| SnapKV | 10% | 100 | 100 | 77.34 | 99.22 | 63.28 | 99.02 | 77.15 | 96.41 | 74.48 | 71.22 | 50 | 82.56 |
| PyramidKV | | 100 | 100 | 71.09 | 100 | 64.84 | 99.81 | 77.15 | 96.41 | 73.44 | 73.5 | 51.56 | 82.53 |
| StreamingLLM | | 6.25 | 11.72 | 11.72 | 9.38 | 10.16 | 13.28 | 12.31 | 12.81 | 94.27 | 31.71 | 32.03 | 22.33 |
| KeyDiff | | 100 | 91.41 | 75 | 82.81 | 4.69 | 73.83 | 58.59 | 96.09 | 70.05 | 38.02 | 31.25 | 65.61 |
| CAKE | | 100 | 100 | 94.53 | 99.22 | 57.03 | 99.61 | 76.38 | 96.56 | 75 | 71.48 | 50.78 | 83.69 |
| CriticalKV | | 100 | 98.44 | 78.91 | 98.44 | 63.28 | 99.41 | 77.54 | 95.16 | 73.7 | 72.01 | 51.56 | 82.59 |
| AnDPro | | 100 | 100 | 91.41 | 99.22 | 76.56 | 99.81 | 76.17 | 96.25 | 74.22 | 72 | 52.34 | 85.27 |
| DropKV | | 100 | 99.22 | 89.94 | 99.22 | 92.97 | 98.83 | 78.13 | 96.88 | 73.7 | 72.01 | 53.13 | **86.73** |
| SnapKV | 30% | 100 | 100 | 100 | 99.22 | 63.91 | 100 | 75 | 96.09 | 76.04 | 71.22 | 53.13 | 84.96 |
| PyramidKV | | 100 | 100 | 98.44 | 99.22 | 66.41 | 100 | 74.81 | 96.25 | 74.22 | 72.2 | 53.91 | 85.04 |
| StreamingLLM | | 28.91 | 35.16 | 26.56 | 26.56 | 26.56 | 31.64 | 28.52 | 35.31 | 93.49 | 31.58 | 30.47 | 35.89 |
| KeyDiff | | 100 | 100 | 100 | 96.09 | 21.09 | 96.48 | 72.07 | 96.56 | 73.18 | 53.39 | 38.28 | 77.01 |
| CAKE | | 100 | 100 | 100 | 99.22 | 48.44 | 99.81 | 75 | 96.41 | 76.82 | 70.7 | 51.56 | 83.45 |
| CriticalKV | | 100 | 100 | 100 | 99.22 | 64.06 | 99.81 | 75 | 94.53 | 77.08 | 71.48 | 53.13 | 84.94 |
| AnDPro | | 100 | 100 | 100 | 99.22 | 61.72 | 99.61 | 75.78 | 96.41 | 76.04 | 71.22 | 53.91 | 84.90 |
| DropKV | | 100 | 100 | 100 | 99.22 | 95.31 | 100 | 75.39 | 96.09 | 76.3 | 71.22 | 53.13 | **87.88** |

*Table 7.* RULER results at 32K sequence length and 5%, 10% and 30% KV-cache budget. The best result is highlighted in gray, and the second-best is underlined.

| Method | Budget | N-S1 | N-S2 | N-S3 | N-MK1 | N-MK2 | N-MQ | N-MV | VT | FWE | QA-1 | QA-2 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 100 | 99.61 | 98.24 | 99.84 | 95.57 | 71.03 | 53.91 | 92.56 |
| SnapKV | 5% | 100 | 99.80 | 48.6 | 99.4 | 97.4 | 99.75 | 98.35 | 98.84 | 81.27 | 68.15 | 55.4 | 86.09 |
| PyramidKV | | 100 | 99.80 | 29 | 100 | 97.4 | 99.75 | 98.2 | 99.24 | 82.6 | 68.85 | 55.6 | 84.59 |
| StreamingLLM | | 5 | 6 | 4.40 | 5.40 | 6.20 | 5.15 | 5.15 | 8.96 | 92.80 | 43.20 | 42.68 | 20.45 |
| KeyDiff | | 100 | 100 | 61.60 | 95 | 2.80 | 91.25 | 91.95 | 98.96 | 65.33 | 42 | 45.50 | 72.22 |
| CAKE | | 100 | 99.60 | 53 | 99.4 | 97.2 | 99.6 | 98.15 | 99.2 | 80.2 | 69.88 | 55.4 | 86.51 |
| CriticalKV | | 100 | 99.80 | 43 | 100 | 97.4 | 99.75 | 97.75 | 99.04 | 76.93 | 68.68 | 55 | 85.21 |
| AnDPro | | 100 | 100.00 | 44.4 | 100 | 99.4 | 99.9 | 98.2 | 99.04 | 76.6 | 68.82 | 54.8 | 85.56 |
| DropKV | | 100 | 100.00 | 84.8 | 100 | 99.6 | 99.6 | 96.8 | 98.72 | 79.33 | 69.25 | 56.8 | **89.54** |
| SnapKV | 10% | 100 | 100 | 91.75 | 100 | 100 | 99.61 | 97.83 | 99.38 | 86.98 | 70.68 | 53.13 | 90.85 |
| PyramidKV | | 100 | 100 | 71.88 | 100 | 100 | 99.81 | 99.02 | 99.53 | 86.72 | 71.29 | 52.34 | 89.14 |
| StreamingLLM | | 14.84 | 10.16 | 13.28 | 13.28 | 9.38 | 10.35 | 10.16 | 17.66 | 95.83 | 44.6 | 42.97 | 25.68 |
| KeyDiff | | 100 | 100 | 94.53 | 100 | 8.59 | 99.61 | 96.29 | 99.53 | 72.92 | 47.59 | 50 | 79.01 |
| CAKE | | 100 | 99.22 | 92.18 | 98.44 | 100 | 99.81 | 98.61 | 99.53 | 84.38 | 69.6 | 54.68 | 90.59 |
| CriticalKV | | 100 | 100 | 89.06 | 100 | 100 | 99.61 | 98.24 | 99.38 | 81.77 | 70.38 | 51.56 | 90.00 |
| AnDPro | | 100 | 100 | 89.84 | 100 | 100 | 99.81 | 98.61 | 99.68 | 85.15 | 69.6 | 53.91 | 90.60 |
| DropKV | | 100 | 100 | 99.22 | 100 | 100 | 99.81 | 97.66 | 99.38 | 82.29 | 69.6 | 53.91 | **91.08** |
| SnapKV | 30% | 100 | 100 | 100 | 100 | 100 | 99.61 | 98.44 | 99.53 | 91.41 | 70.64 | 53.13 | 92.07 |
| PyramidKV | | 100 | 100 | 100 | 100 | 100 | 99.61 | 98.44 | 99.53 | 90.63 | 70.51 | 53.13 | 91.99 |
| StreamingLLM | | 30.47 | 37.5 | 35.16 | 35.16 | 34.38 | 31.64 | 33.59 | 41.72 | 96.09 | 41.99 | 45.31 | 42.09 |
| KeyDiff | | 100 | 100 | 100 | 100 | 38.28 | 99.22 | 98.05 | 99.53 | 87.76 | 61.85 | 54.69 | 85.04 |
| CAKE | | 100 | 99.22 | 100 | 98.44 | 100 | 99.81 | 99.22 | 99.53 | 91.67 | 69.2 | 52.34 | 91.77 |
| CriticalKV | | 100 | 100 | 100 | 100 | 100 | 99.41 | 98.44 | 99.53 | 90.63 | 69.88 | 53.91 | 91.98 |
| AnDPro | | 100 | 100 | 100 | 100 | 100 | 99.61 | 98.05 | 99.53 | 90.67 | 70 | 53.91 | 91.98 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 99.41 | 98.63 | 99.53 | 90.76 | 70 | 54.69 | **92.09** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 100 | 100 | 94.34 | 97.34 | 95.57 | 62.63 | 55.47 | 91.40 |
| SnapKV | 5% | 100 | 100 | 2.8 | 100 | 82 | 99.9 | 86.45 | 97.92 | 89.33 | 62.85 | 55.8 | 79.73 |
| PyramidKV | | 100 | 100 | 0.6 | 100 | 62.2 | 98.85 | 83.1 | 87.76 | 76.67 | 61.65 | 54.2 | 75.00 |
| StreamingLLM | | 5 | 4.80 | 5 | 5.40 | 3 | 4.55 | 5.65 | 8.96 | 87.80 | 38.60 | 32.67 | 18.31 |
| KeyDiff | | 100 | 99.80 | 98.40 | 86.40 | 0.20 | 73.15 | 68.40 | 94.60 | 86.07 | 24 | 30.53 | 69.23 |
| CAKE | | 100 | 99.8 | 3.6 | 99.4 | 15.2 | 97.65 | 79.3 | 98 | 89.13 | 62.73 | 55.8 | 72.78 |
| CriticalKV | | 100 | 100 | 4 | 100 | 88.6 | 99.9 | 86.6 | 97.92 | 90 | 62.78 | 55.6 | 80.49 |
| AnDPro | | 99.8 | 99.8 | 20.4 | 100 | 87 | 99.95 | 89.6 | 98.44 | 88.2 | 63.05 | 55.8 | 82.00 |
| DropKV | | 100 | 99.6 | 63.8 | 100 | 99 | 99.8 | 90 | 98.2 | 90.8 | 62.95 | 56.6 | **87.34** |
| SnapKV | 10% | 100 | 100 | 14.06 | 100 | 92.19 | 100 | 87.11 | 97.81 | 91.93 | 62.89 | 53.91 | 81.81 |
| PyramidKV | | 100 | 100 | 1.56 | 100 | 82.03 | 100 | 88.09 | 94.84 | 88.8 | 62.63 | 55.47 | 79.40 |
| StreamingLLM | | 14.06 | 14.06 | 10.94 | 9.38 | 9.38 | 10.55 | 12.31 | 15.58 | 90.63 | 31.58 | 38.28 | 23.34 |
| KeyDiff | | 100 | 99.22 | 30.47 | 98.44 | 3.13 | 95.7 | 84.96 | 97.81 | 90.63 | 37.57 | 32.03 | 70.00 |
| CAKE | | 100 | 100 | 20.31 | 100 | 28.91 | 100 | 87.5 | 97.97 | 92.19 | 62.37 | 55.47 | 76.79 |
| CriticalKV | | 100 | 100 | 23.44 | 100 | 95.31 | 100 | 87.11 | 98.13 | 93.49 | 62.83 | 54.69 | 83.18 |
| AnDPro | | 100 | 100 | 64.84 | 100 | 96.88 | 100 | 89.84 | 98.59 | 90.89 | 64.97 | 54.68 | 87.34 |
| DropKV | | 100 | 100 | 85.94 | 100 | 100 | 100 | 90.82 | 98.44 | 93.75 | 63.93 | 56.25 | **89.92** |
| SnapKV | 30% | 100 | 100 | 71.09 | 100 | 97.66 | 100 | 88.09 | 97.5 | 95.57 | 62.37 | 55.47 | 87.98 |
| PyramidKV | | 100 | 100 | 15.63 | 100 | 92.19 | 100 | 91.99 | 97.81 | 93.75 | 62.63 | 56.25 | 82.75 |
| StreamingLLM | | 25 | 35.94 | 31.25 | 39.06 | 36.72 | 38.09 | 35.35 | 43.13 | 91.15 | 27.6 | 42.19 | 40.50 |
| KeyDiff | | 100 | 100 | 100 | 100 | 26.56 | 99.81 | 92.38 | 96.88 | 96.35 | 53.52 | 49.22 | 83.16 |
| CAKE | | 100 | 100 | 89.84 | 100 | 67.97 | 100 | 90.82 | 97.19 | 94.79 | 62.37 | 55.47 | 87.13 |
| CriticalKV | | 100 | 100 | 87.5 | 100 | 97.66 | 100 | 90.82 | 97.97 | 95.57 | 64.19 | 55.47 | 89.93 |
| AnDPro | | 100 | 100 | 95.31 | 100 | 100 | 100 | 89.45 | 97.97 | 93.23 | 62.13 | 55.25 | 90.30 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 100 | 91.21 | 97.81 | 95.31 | 62.63 | 55.47 | **91.13** |
| **Mistral-7B-512K** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 100 | 99.81 | 78.32 | 98.44 | 96.09 | 77.54 | 49.22 | 90.86 |
| SnapKV | 5% | 100 | 100 | 25 | 99.6 | 87.6 | 99.05 | 77.05 | 97.68 | 86.4 | 72.9 | 52 | 81.57 |
| PyramidKV | | 100 | 100 | 13.2 | 99.4 | 86.4 | 99.15 | 78.05 | 96.72 | 86.87 | 72.3 | 51 | 80.28 |
| StreamingLLM | | 4.60 | 6.60 | 5.60 | 6.40 | 5.40 | 5.80 | 7 | 6.56 | 83.40 | 34.40 | 37.32 | 18.46 |
| KeyDiff | | 100 | 75.40 | 10 | 49.60 | 0.80 | 28.50 | 31.45 | 97.16 | 74 | 23 | 38.77 | 48.06 |
| CAKE | | 100 | 100 | 47.7 | 99.4 | 72.8 | 99.1 | 76.8 | 97.2 | 87.27 | 73.43 | 52 | 82.34 |
| CriticalKV | | 100 | 100 | 20.6 | 99.4 | 86 | 99.05 | 77.3 | 97.44 | 85.4 | 72.9 | 51.2 | 80.84 |
| AnDPro | | 100 | 99.6 | 36.6 | 99.6 | 95.4 | 98.45 | 81.05 | 97.48 | 86.88 | 73.77 | 51.4 | 83.86 |
| DropKV | | 100 | 99.8 | 46.8 | 99.6 | 99.4 | 97.68 | 79.45 | 97.72 | 87.6 | 74.17 | 50.2 | **84.77** |
| SnapKV | 10% | 100 | 100 | 57.81 | 100 | 89.84 | 99.61 | 79.3 | 98.13 | 89.06 | 75.13 | 47.66 | 85.14 |
| PyramidKV | | 100 | 100 | 50 | 100 | 87.5 | 99.61 | 80.08 | 98.28 | 87.76 | 76.38 | 47.66 | 84.30 |
| StreamingLLM | | 6.25 | 15.63 | 14.84 | 14.84 | 11.72 | 13.28 | 14.06 | 14.22 | 92.19 | 29.82 | 32.81 | 23.61 |
| KeyDiff | | 100 | 92.97 | 55.47 | 78.91 | 3.13 | 72.66 | 55.47 | 97.66 | 80.73 | 43.1 | 25.78 | 64.17 |
| CAKE | | 100 | 100 | 90.63 | 100 | 77.34 | 99.81 | 78.91 | 97.66 | 91.15 | 76.69 | 48.44 | 87.33 |
| CriticalKV | | 100 | 100 | 65.63 | 100 | 87.5 | 99.61 | 80.27 | 97.66 | 88.8 | 76.17 | 47.66 | 85.75 |
| AnDPro | | 100 | 100 | 73.44 | 100 | 92.18 | 99.61 | 81.05 | 97.66 | 91.41 | 74.61 | 47.66 | 87.03 |
| DropKV | | 100 | 100 | 79.69 | 100 | 99.22 | 98.83 | 81.25 | 97.81 | 91.41 | 76.43 | 46.88 | **88.32** |
| SnapKV | 30% | 100 | 100 | 98.44 | 100 | 85.94 | 99.81 | 79.88 | 98.44 | 92.45 | 76.95 | 48.44 | 89.12 |
| PyramidKV | | 100 | 100 | 97.66 | 100 | 90.63 | 99.81 | 79.88 | 98.91 | 91.15 | 77.47 | 49.22 | 89.52 |
| StreamingLLM | | 23.44 | 35.16 | 33.59 | 35.16 | 33.59 | 30.67 | 33.01 | 40.78 | 89.06 | 30.86 | 35.94 | 38.30 |
| KeyDiff | | 100 | 99.22 | 99.22 | 96.09 | 20.31 | 96.29 | 72.85 | 97.66 | 88.8 | 63.28 | 38.28 | 79.27 |
| CAKE | | 100 | 100 | 100 | 100 | 82.81 | 99.81 | 78.13 | 98.44 | 94.53 | 77.54 | 49.22 | 89.13 |
| CriticalKV | | 100 | 100 | 99.22 | 100 | 89.6 | 99.81 | 80.27 | 98.13 | 94.27 | 76.95 | 49.22 | 89.77 |
| AnDPro | | 100 | 100 | 99.22 | 100 | 89.06 | 99.81 | 80.27 | 98.44 | 94.01 | 76.95 | 49.22 | 89.73 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 99.61 | 79.88 | 98.44 | 94.27 | 77.28 | 48.44 | **90.72** |

*Table 8.* RULER results at 16K sequence length and 5%, 10% and 30% KV-cache budget. The best result is highlighted in gray, and the second-best is underlined.

| **Method** | Budget | N-S1 | N-S2 | N-S3 | N-MK1 | N-MK2 | N-MQ | N-MV | VT | FWE | QA-1 | QA-2 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LLaMA-3.1-8B-Instruct** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 100 | 100 | 98.05 | 99.84 | 96.35 | 76.3 | 59.38 | 93.63 |
| SnapKV | 5% | 100 | 100.00 | 19.92 | 100 | 98.44 | 99.9 | 97.56 | 99.06 | 91.28 | 71.62 | 58.59 | 85.12 |
| PyramidKV | | 100 | 100.00 | 7.81 | 100 | 96.88 | 99.61 | 98.63 | 99.14 | 91.67 | 67.45 | 58.2 | 83.58 |
| StreamingLLM | | 3.80 | 6.20 | 5 | 4.80 | 5.40 | 4 | 5.95 | 8.04 | 91.73 | 43.40 | 44.12 | 20.22 |
| KeyDiff | | 100 | 100 | 49.60 | 93.60 | 3 | 89.15 | 89.20 | 98.88 | 83.67 | 44.80 | 45.25 | 72.47 |
| CAKE | | 100 | 100.00 | 21.88 | 99.61 | 97.27 | 99.71 | 97.66 | 99.22 | 91.15 | 67.77 | 58.98 | 84.84 |
| CriticalKV | | 100 | 100.00 | 19.53 | 100 | 100 | 99.81 | 97.36 | 99.38 | 90.1 | 68.88 | 57.81 | 84.81 |
| AnDPro | | 100 | 100.00 | 24.61 | 100 | 98.82 | 100 | 98.44 | 98.98 | 91.28 | 68.95 | 57.81 | 85.35 |
| DropKV | | 100 | 100.00 | 49.61 | 100 | 99.61 | 99.02 | 97.36 | 98.83 | 93.1 | 69.14 | 57.81 | **87.68** |
| SnapKV | 10% | 100 | 100 | 71.09 | 100 | 99.22 | 100 | 98.44 | 99.53 | 94.53 | 76.04 | 59.38 | 90.75 |
| PyramidKV | | 100 | 100 | 38.28 | 100 | 100 | 100 | 99.02 | 99.84 | 95.05 | 74.74 | 59.38 | 87.85 |
| StreamingLLM | | 6.25 | 11.72 | 14.87 | 6.25 | 11.72 | 8.39 | 9.77 | 14.49 | 92.19 | 45.64 | 43.75 | 24.09 |
| KeyDiff | | 100 | 100 | 96.09 | 100 | 10.94 | 100 | 96.88 | 99.53 | 85.16 | 51.37 | 49.22 | 80.84 |
| CAKE | | 100 | 100 | 71.09 | 100 | 99.22 | 99.81 | 99.02 | 99.69 | 92.71 | 75.26 | 59.38 | 90.56 |
| CriticalKV | | 100 | 100 | 61.72 | 100 | 100 | 100 | 98.24 | 99.69 | 94.53 | 75.26 | 60.15 | 89.96 |
| AnDPro | | 100 | 100 | 71.88 | 100 | 100 | 100 | 98.63 | 99.69 | 94.53 | 75.85 | 59.38 | 90.91 |
| DropKV | | 100 | 100 | 81.25 | 100 | 100 | 99.61 | 98.05 | 99.69 | 94.27 | 75.26 | 59.38 | **91.59** |
| SnapKV | 30% | 100 | 100 | 100 | 100 | 100 | 100 | 98.41 | 99.84 | 96.41 | 74.48 | 60.14 | 93.57 |
| PyramidKV | | 100 | 100 | 98.44 | 100 | 100 | 100 | 98.63 | 98.44 | 96.35 | 73.11 | 60.16 | 93.19 |
| StreamingLLM | | 22.66 | 32.03 | 39.84 | 30.47 | 33.59 | 30.27 | 33.79 | 40.78 | 91.93 | 74.74 | 40.78 | 42.81 |
| KeyDiff | | 100 | 100 | 100 | 100 | 46.09 | 100 | 98.05 | 99.84 | 94.1 | 59.11 | 57.03 | 86.75 |
| CAKE | | 100 | 99.22 | 100 | 100 | 99.22 | 100 | 99.22 | 99.84 | 96.35 | 73.96 | 58.59 | 93.31 |
| CriticalKV | | 100 | 100 | 100 | 100 | 100 | 100 | 98.02 | 99.84 | 96.88 | 74.74 | 59.38 | 93.53 |
| AnDPro | | 100 | 100 | 100 | 100 | 100 | 100 | 97.74 | 99.84 | 96.03 | 75 | 60.23 | 93.53 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 100 | 97.85 | 99.84 | 96.09 | 75.59 | 60.16 | **93.59** |
| **Qwen2.5-7B-Instruct-1M** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 99.22 | 100 | 92.38 | 99.53 | 95.31 | 66.34 | 66.41 | 92.65 |
| SnapKV | 5% | 100 | 99.61 | 1.17 | 100 | 70.31 | 99.71 | 84.08 | 99.53 | 91.93 | 60.55 | 57.03 | 78.54 |
| PyramidKV | | 100 | 99.61 | 0 | 100 | 36.72 | 87.99 | 59.77 | 68.52 | 79.95 | 58.59 | 50.78 | 67.54 |
| StreamingLLM | | 5.60 | 5.40 | 4 | 7 | 4.20 | 5.20 | 4.90 | 7.84 | 85.53 | 40.80 | 33.28 | 18.52 |
| KeyDiff | | 100 | 99.60 | 93.40 | 70.60 | 0.80 | 58.70 | 56.65 | 96.60 | 83.40 | 18.80 | 24.32 | 63.90 |
| CAKE | | 100 | 99.61 | 1.5 | 97.66 | 8.59 | 94.04 | 65.53 | 99.3 | 92.58 | 59.9 | 57.42 | 70.56 |
| CriticalKV | | 100 | 99.61 | 2.34 | 100 | 87.5 | 99.51 | 85.45 | 99.45 | 92.06 | 60.55 | 57.42 | 80.35 |
| AnDPro | | 100 | 99.61 | 11.72 | 100 | 76.18 | 99.9 | 88.67 | 99.22 | 92.58 | 61.78 | 57.03 | 80.61 |
| DropKV | | 100 | 99.22 | 51.17 | 100 | 100 | 99.81 | 88.28 | 99.45 | 91.41 | 62.89 | 56.64 | **86.26** |
| SnapKV | 10% | 100 | 100 | 7.8 | 100 | 85.94 | 99.41 | 87.7 | 99.69 | 92.45 | 66.41 | 62.5 | 81.99 |
| PyramidKV | | 100 | 100 | 1.56 | 100 | 62.5 | 98.05 | 83.2 | 92.5 | 88.02 | 68.23 | 63.28 | 77.94 |
| StreamingLLM | | 10.94 | 17.97 | 9.38 | 12.5 | 7.03 | 10.35 | 11.13 | 14.69 | 88.28 | 30.66 | 37.5 | 22.77 |
| KeyDiff | | 100 | 100 | 100 | 95.31 | 3.91 | 88.48 | 80.47 | 97.97 | 84.9 | 26.63 | 31.25 | 73.54 |
| CAKE | | 100 | 100 | 12.5 | 100 | 21.09 | 98.83 | 83.59 | 99.84 | 91.93 | 66.67 | 64.06 | 76.23 |
| CriticalKV | | 100 | 100 | 14.06 | 100 | 91.41 | 100 | 88.87 | 100 | 94.01 | 68.49 | 66.41 | 83.93 |
| AnDPro | | 100 | 100 | 38.28 | 100 | 85.16 | 100 | 90.04 | 99.84 | 94.01 | 67.19 | 65.63 | 85.47 |
| DropKV | | 100 | 100 | 88.28 | 100 | 100 | 100 | 89.26 | 99.84 | 93.75 | 68.75 | 64.84 | **91.34** |
| SnapKV | 30% | 100 | 100 | 52.34 | 100 | 96.88 | 100 | 88.09 | 99.53 | 96.35 | 66.67 | 65.63 | 87.77 |
| PyramidKV | | 100 | 100 | 10.16 | 100 | 85.16 | 99.81 | 89.65 | 99.69 | 94.27 | 66.15 | 66.41 | 82.85 |
| StreamingLLM | | 27.34 | 35.94 | 34.38 | 35.16 | 22.66 | 33.98 | 32.81 | 40.94 | 90.89 | 67.45 | 42.19 | 42.16 |
| KeyDiff | | 100 | 100 | 100 | 100 | 21.09 | 100 | 92.19 | 99.38 | 90.89 | 46.75 | 50.78 | 81.92 |
| CAKE | | 100 | 100 | 67.19 | 100 | 55.47 | 100 | 88.87 | 99.84 | 96.09 | 64.78 | 66.41 | 85.33 |
| CriticalKV | | 100 | 100 | 75 | 100 | 98.44 | 100 | 89.06 | 99.69 | 96.62 | 66.15 | 67.19 | 90.20 |
| AnDPro | | 100 | 100 | 83.59 | 100 | 96.09 | 100 | 88.28 | 100 | 96.88 | 66.12 | 66.41 | 90.67 |
| DropKV | | 100 | 100 | 100 | 100 | 100 | 100 | 89.06 | 100 | 96.09 | 66.15 | 64.06 | **92.31** |
| **Mistral-7B-512K** | | | | | | | | | | | | | |
| Full KV | 100% | 100 | 100 | 100 | 100 | 100 | 99.61 | 85.35 | 98.28 | 87.24 | 75.13 | 57.03 | 91.15 |
| SnapKV | 5% | 100 | 100 | 14.45 | 99.22 | 87.11 | 98.14 | 72.27 | 95.94 | 84.51 | 69.6 | 53.13 | 79.49 |
| PyramidKV | | 100 | 100 | 6.64 | 99.22 | 87.89 | 97.95 | 74.9 | 93.91 | 83.59 | 68 | 50.39 | 78.41 |
| StreamingLLM | | 4.20 | 5.20 | 5.80 | 4.20 | 4.60 | 4.65 | 2.80 | 6.72 | 84.93 | 33.80 | 39.48 | 17.85 |
| KeyDiff | | 100 | 75.80 | 2.60 | 43.60 | 0.20 | 24.80 | 25.95 | 96.16 | 70.80 | 21.20 | 31.63 | 44.79 |
| CAKE | | 99.61 | 99.61 | 26.95 | 98.44 | 80.08 | 98.54 | 74.61 | 95.55 | 84.38 | 68.16 | 52.73 | 79.88 |
| CriticalKV | | 100 | 99.61 | 12.89 | 99.22 | 85.55 | 97.75 | 75.29 | 95.78 | 82.81 | 67.87 | 52.73 | 79.05 |
| AnDPro | | 100 | 100 | 24.61 | 99.61 | 96.88 | 98.61 | 85.06 | 95.08 | 86.46 | 69.25 | 52.34 | 82.54 |
| DropKV | | 100 | 99.61 | 31.64 | 98.83 | 99.61 | 95.51 | 83.69 | 95.86 | 84.25 | 67.38 | 52.73 | **82.65** |
| SnapKV | 10% | 100 | 100 | 48.44 | 100 | 90.63 | 99.61 | 83.4 | 98.13 | 84.38 | 75.13 | 53.13 | 84.80 |
| PyramidKV | | 100 | 100 | 36.72 | 100 | 89.84 | 99.41 | 83.79 | 95.78 | 83.85 | 75.07 | 53.13 | 83.42 |
| StreamingLLM | | 12.5 | 16.41 | 13.28 | 13.28 | 11.72 | 12.89 | 8.01 | 17.5 | 87.76 | 37.31 | 35.16 | 24.17 |
| KeyDiff | | 100 | 85.94 | 34.38 | 71.09 | 0.78 | 60.35 | 45.12 | 98.44 | 73.7 | 42.58 | 26.56 | 58.09 |
| CAKE | | 100 | 100 | 72.66 | 98.44 | 82.03 | 99.81 | 83.98 | 98.12 | 84.64 | 74.09 | 54.69 | 86.22 |
| CriticalKV | | 100 | 100 | 43.75 | 100 | 92.19 | 99.02 | 85.35 | 97.66 | 83.59 | 73.83 | 53.91 | 84.48 |
| AnDPro | | 100 | 100 | 62.5 | 100 | 97.66 | 99.61 | 87.11 | 97.5 | 86.2 | 73.56 | 53.12 | 87.02 |
| DropKV | | 100 | 100 | 64.84 | 100 | 100 | 99.22 | 86.13 | 98.28 | 84.12 | 75.46 | 53.91 | **87.45** |
| SnapKV | 30% | 100 | 100 | 95.31 | 100 | 91.41 | 100 | 85.55 | 98.13 | 84.64 | 74.35 | 55.47 | 89.53 |
| PyramidKV | | 100 | 100 | 91.41 | 100 | 92.19 | 100 | 85.16 | 98.28 | 83.85 | 75.39 | 56.25 | 89.32 |
| StreamingLLM | | 31.25 | 25 | 31.25 | 32.81 | 34.38 | 33.2 | 25.2 | 44.06 | 86.98 | 31.84 | 40.63 | 37.87 |
| KeyDiff | | 100 | 100 | 97.66 | 92.97 | 21.09 | 94.53 | 76.56 | 97.97 | 85.42 | 57.55 | 41.41 | 78.65 |
| CAKE | | 100 | 100 | 99.22 | 100 | 89.06 | 100 | 83.98 | 98.12 | 85.16 | 74.35 | 56.25 | 89.65 |
| CriticalKV | | 100 | 100 | 95.31 | 100 | 93.75 | 100 | 85.74 | 97.34 | 85.16 | 74.35 | 56.25 | 89.81 |
| AnDPro | | 100 | 100 | 97.22 | 100 | 96.09 | 99.61 | 85.55 | 97.34 | 85.94 | 74.34 | 56.25 | 90.21 |
| DropKV | | 100 | 100 | 96.88 | 100 | 100 | 100 | 87.11 | 98.28 | 85.16 | 75.72 | 55.47 | **90.78** |

### A.1. Serving Throughput vs. Dynamic Cache

**Serving throughput vs. dynamic cache.** As shown in Table 9, with DropKV the maximum batch ranges from $1.75\times$ (Qwen, $n$=128K) to $6.87\times$ (Mistral, $n$=128K) of the dynamic baseline, with Triton and PyTorch DropKV realizing comparable capacity on Llama, a clear Triton lead on Mistral at every $n$ (peaking at $B$=3 vs. $B$=2 and $70.8\rightarrow86.5$ tok/s at $n$=128K), and Triton matched-batch-neutral on Qwen only at $n$=32K ($\delta=-0.1\%$) and positive at $n$=4K ($+1.9\%$) and $n$=128K ($+3.3\%$); Qwen's smaller $H_{\text{kv}}$=4 halves the kernel grid $B{\cdot}H_{\text{kv}}$, dominant at intermediate $n$.

*Table 9.* Serving throughput at each method's maximum batch at $5\%$ budget on A100-80GB GPU. $B$ is the largest batch that fits for the given $(n, \text{method})$; aggregate decode throughput is in tokens/s. $\times_{\text{dyn}}$ is Triton's decode throughput divided by dynamic's.

| | | Dynamic | | DropKV-PyTorch | | DropKV-Triton | | |
|---|---|---|---|---|---|---|---|---|
| Model | $n$ | $B$ | tok/s | $B$ | tok/s | $B$ | tok/s | $\times_{\text{dyn}}$ |
| Llama-3.1-8B | 4k | 38 | 395 | 56 | 2111 | 56 | 2066 | 5.23 |
| | 32k | 4 | 50 | 7 | 260 | 6 | 223 | 4.46 |
| | 128k | 1 | 12 | 1 | 38 | 1 | 38 | 3.02 |
| Mistral-7B-512k | 4k | 56 | 411 | 88 | 2805 | 99 | 2938 | 7.15 |
| | 32k | 7 | 55 | 11 | 413 | 12 | 452 | 8.22 |
| | 128k | 1 | 13 | 2 | 71 | 3 | 86 | 6.87 |
| Qwen2.5-7B-1M | 4k | 42 | 793 | 52 | 2169 | 51 | 2141 | 2.70 |
| | 32k | 5 | 100 | 6 | 246 | 6 | 246 | 2.46 |
| | 128k | 1 | 23 | 1 | 41 | 1 | 41 | 1.75 |

### A.2. Time to First Token latency

Table 10 reports Time to First Tokenm (TTFT) latency on Mistral-7B-512K with 32K seq. length, 500 decoding steps, 5% budget. Among eviction methods, DropKV consistently delivers the lowest total TTFT at every batch size. Its TTFT also matches the dynamic baseline within $0.6\%$ where Dynamic is feasible, indicating that the prefill eviction overhead remains minimal while the compressed cache improves decoding efficiency.

*Table 10.* Time to First Token latencies (in seconds) across batch sizes $B \in \{1, 2, 4, 8, 12\}$. All evaluations were conducted using the Mistral-7B-512K model with 32K sequence length. ↓ indicates lower values are better. "OOM" denotes Out-of-Memory.

| Method | Time to First Token (s) ↓ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 12 |
| Dynamic | 3.69 | **7.36** | 14.83 | OOM | OOM |
| SnapKV | 3.95 | 7.47 | 14.92 | 30.36 | 46.38 |
| CriticalKV | 4.24 | 8.27 | 16.68 | 33.62 | 51.96 |
| AnDPro | 5.31 | 10.30 | 20.45 | 41.00 | 63.43 |
| **DropKV** | **3.68** | 7.40 | **14.75** | **29.94** | **46.29** |

## B. Scoring Functions of Prior Eviction Methods

The eviction baselines compared in Table 1 can all be cast as surrogates of the same NP-hard problem $\min_{\mathcal{J}} \|\Delta(\mathcal{J})\|_2$, but they differ in *what* surrogate they construct: (a) *heuristic retention scores* that depend only on cache-side proxies (position, key vectors, or attention magnitudes) and do not model the value-space output perturbation, and (b) *output-perturbation surrogates* that explicitly target $\|\Delta(\mathcal{J})\|_2$ via different relaxations of the combinatorial objective. Throughout this section, $\mathcal{Q}$ denotes the SnapKV-style observation window of $w$ recent queries, $p_j^{(t)}$ the softmax attention weight from query $\boldsymbol{q}^{(t)} \in \mathcal{Q}$ to token $j$, $\boldsymbol{a}^{(t)} = \boldsymbol{V}^\top \boldsymbol{p}^{(t)}$ the corresponding pre-eviction attention output, $\boldsymbol{W}_O$ the per-head output projection, and $\bar{p}_j = \frac{1}{|\mathcal{Q}|}\sum_{t\in\mathcal{Q}} p_j^{(t)}$. All baseline scores below are *retention* scores (large = keep); DropKV's score is an *eviction cost* (small = evict). Table 11 summarises the per-token expressions in one line per method.

*Table 11.* Per-token scoring expressions of the baselines in Table 1. "Type" classifies the surrogate strategy: **H:** heuristic retention (no value-space modelling), **B:** explicit upper bound on $\|\Delta(\mathcal{J})\|_1$, **R:** convex (LASSO-style (Tibshirani, 1996)) relaxation of the combinatorial perturbation objective, **E:** exact singleton perturbation followed by per-token decoupling. "Direction" indicates whether the score is read as "keep large" (retention score) or "evict small" (eviction cost). Notation: $\bar{p}_j = \frac{1}{|\mathcal{Q}|}\sum_{t\in\mathcal{Q}} p_j^{(t)}$; $\mu(\boldsymbol{K}) = \frac{1}{n}\sum_i \boldsymbol{k}_i$; $s$, $w$ are the StreamingLLM sink and recent-window sizes; $\kappa$ is the SnapKV pooling kernel size; $\gamma$ is CAKE's variance weight; and $k^l$ is PyramidKV's per-layer budget.

| Method | Type | Per-token score | Direction | Reference |
|---|---|---|---|---|
| StreamingLLM (Xiao et al., 2023) | H | $\mathbb{1}[j \le s] + \mathbb{1}[j > n-w]$ | keep large (mask) | §3.2 |
| KeyDiff (Park et al., 2025) | H | $-\operatorname{CosSim}\big(\boldsymbol{k}_j,\ \mu(\boldsymbol{K})\big)$ | keep large | Eq. 8 |
| SnapKV (Li et al., 2024) | H | $\operatorname{MaxPool1d}_\kappa\big(\sum_{t\in\mathcal{Q}} p_j^{(t)}\big)$ | keep large | Eq. 2–3 |
| PyramidKV (Cai et al., 2024) | H | SnapKV per-token; layer-asym. budget $k^l$ | keep large | Eq. 1, 3 |
| CAKE (Qin et al., 2025) | H | $\operatorname{Mean}(\boldsymbol{A}[-w{:},j]) + \gamma\operatorname{Var}(\boldsymbol{A}[-w{:},j])$ | keep large | Eq. 11 |
| CriticalKV (Feng et al., 2025) | B | $(\bar{p}_j+\epsilon)\,\|(\boldsymbol{V}\boldsymbol{W}_O)_{j,:}\|_1$ | keep large | Alg. 1; Thm. 3.5 |
| AnDPro (Geng et al., 2025) | R | $\sum_{t\in\mathcal{Q}} p_j^{(t)}\ \langle \boldsymbol{a}^{(t)},\ \boldsymbol{v}_j\rangle$ | keep large | Eq. 15 |
| **DropKV** (ours) | E | $\sum_{t\in\mathcal{Q}}\big(p_j^{(t)}/(1-p_j^{(t)})\big)^2\,\|\boldsymbol{a}^{(t)}-\boldsymbol{v}_j\|_2^2$ | evict small | Lem. 1 |

**Value-aware surrogates.** CriticalKV minimizes an explicit upper bound $\widehat{\theta}$ on the $\ell_1$ analog of $\|\Delta(\mathcal{J})\|_2$ after the per-head output projection $\boldsymbol{W}_O$ ((Feng et al., 2025), Theorems 3.3 and 3.5); the listed score is the stage-2 criterion of a two-stage selection in which a fraction $\alpha$ of the cache is filled by raw $\bar{p}_j$ first and the remainder by the composite score, with $\alpha=0.5$ in the paper's experiments (the listed default in Algorithm 1 is $\alpha=0.25$). AnDPro starts from the value-space reconstruction problem $\min_{\boldsymbol{z}\in\{0,1\}^n}\frac{1}{2}\|\boldsymbol{y}-\frac{\sum_i z_i a_i \boldsymbol{v}_i}{\sum_i z_i a_i}\|_2^2$, relaxes the binary mask to $\beta_i\in[0,1/s]$ with a LASSO penalty (Problem P1 of (Geng et al., 2025), §3.2), and uses the KKT screening condition with anchor $\theta=\boldsymbol{y}^{(t)}$ to derive the per-token retention score $p_j^{(t)}\langle \boldsymbol{a}^{(t)},\boldsymbol{v}_j\rangle$ (Eq. 13–15). DropKV instead uses the *exact* per-token eviction perturbation $\|\Delta(\{j\})\|_2 = \frac{p_j}{1-p_j}\|\boldsymbol{a}-\boldsymbol{v}_j\|_2$ from Lemma 1 as the per-token cost and decouples the multi-token objective into a window-summed sum of squared tokens; the decoupling is justified by the cone condition (Theorem 1), which we verify empirically at deployment scale (Appendix C.3).

**SnapKV as the attention-only baseline.** Since all methods use the same observation window $w=8$ and pooling kernel $\kappa=11$ (§5), the SnapKV vs. DropKV gap in Table 1 & 5 directly isolates the contribution of the value-residual factor $\|\boldsymbol{a}^{(t)}-\boldsymbol{v}_j\|_2$ over attention-weighted scoring: $+2.43$, $+7.68$, and $+7.09$ average RULER points on Llama-3.1-8B, Mistral-7B-512K, and Qwen2.5-7B-1M at $n=128$K, $5\%$ budget. The two scores differ only in pool order (SnapKV pools after summing attention; DropKV pools after summing the squared score), which is negligible at the small attention weights typical of the eviction pool.

**Pre- vs. post-$\boldsymbol{W}_O$ scoring.** DropKV scores tokens by the pre-projection perturbation $\|\boldsymbol{a}^{(t)}-\boldsymbol{v}_j\|_2$ rather than the residual-stream perturbation $\|\boldsymbol{W}_O^{(h)}(\boldsymbol{a}^{(t)}-\boldsymbol{v}_j)\|_2$ that the rest of the network actually receives. By submultiplicativity, $\|\boldsymbol{W}_O^{(h)}\delta\|_2 \le \|\boldsymbol{W}_O^{(h)}\|_{\mathrm{op}}\|\delta\|_2$; when the head projection is well-conditioned on the perturbation directions, the pre- and post-projection objectives differ by at most the corresponding conditioning factor, leaving the form of Theorem 1 unchanged while changing only this constant. Empirically, the closest baseline that explicitly uses $\boldsymbol{W}_O$-projected value representations is CriticalKV (Feng et al., 2025), whose stage-2 score ranks tokens by attention weight times the $L_1$ norm of $\boldsymbol{W}_O$-projected values; while this comparison also changes the norm, attention weighting, and selection procedure, DropKV — which omits the projection — outperforms CriticalKV at $n=128$K and $5\%$ budget by $+3.28$, $+7.34$, and $+6.34$ average RULER points on Llama-3.1-8B, Mistral-7B-512K, and Qwen2.5-7B-1M, respectively (Table 5), suggesting explicit output projection is not necessary at the score level.

**DropKV vs. AnDPro at $\boldsymbol{v}_j \approx \boldsymbol{a}$.** DropKV and AnDPro target the same value-space output but their per-token surrogates disagree in sign within a structurally important regime. By Lemma 1, removing a single token $j$ shifts the output by $\Delta(\{j\}) = \frac{p_j}{1-p_j}(\boldsymbol{a}-\boldsymbol{v}_j)$, so a token with $\boldsymbol{v}_j \approx \boldsymbol{a}$ produces a near-zero shift and is therefore safely evictable. DropKV correctly assigns it a near-zero eviction cost. AnDPro's KKT-derived score $p_j\langle \boldsymbol{a},\boldsymbol{v}_j\rangle$, in contrast, is large precisely when

$\boldsymbol{v}_j$ aligns with $\boldsymbol{a}$, so AnDPro *retains* the same token. The two surrogates approximate the same combinatorial objective $\|\Delta(\mathcal{J})\|_2$ but disagree pointwise at strong $\boldsymbol{v}_j$-$\boldsymbol{a}$ alignment, which is the regime projection-based scoring is designed to reward. On RULER at the $5\%$ budget, DropKV improves average accuracy over AnDPro by $+2.35$, $+3.87$, and $+4.03$ points on Llama-3.1-8B-Instruct, Qwen2.5-7B-1M, and MegaBeam-Mistral-7B-512K respectively (Table 5), consistent with the per-token perturbation surrogate being the more faithful one.

**Linear vs. quadratic attention weighting.** CriticalKV and AnDPro weight scores by $p_j$ linearly; DropKV uses $(p_j/(1-p_j))^2$. At deployment-scale eviction-pool weights $p_j \sim 7.6\times10^{-6}$ (uniform $1/n$ at $n = 128$K), the $1/(1-p_j)$ correction contributes at most $\approx 1.5\times10^{-5}$ relative deviation; the substantive gap between DropKV and the prior value-aware surrogates is therefore between linear and quadratic in $p_j$, which polarizes DropKV's eviction cost toward the few less-tiny attention weights when they are present in the candidate pool $\mathcal{L}$.

## C. Theoretical Analysis

### C.1. NP-hardness of Exact Eviction Minimization

Even under the small-weights regime in Assumption 1, exact minimization of $F(\mathcal{J}) = \|\Delta(\mathcal{J})\|_2$ is NP-hard. The obstacle is the numerator $\|\sum_{j\in\mathcal{J}} p_j \boldsymbol{r}_j\|_2$, which is the norm of a sum couples all elements of $\mathcal{J}$, so the objective is not decomposable over individual tokens.

To see this, consider the scalar case $d = 1$. Given an instance of **Balanced Partition** (Hartmanis, 1982), namely $2m$ positive integers $a_1, \ldots, a_{2m}$ with total sum $A$, decide whether they can be split into two equal-size groups of equal sum. Set $n = 2m$, $k = m$, $p_j = 1/n$ for all $j \in [n]$, and $\boldsymbol{v}_j = a_j$, $\boldsymbol{a} = A/n$, so that $\boldsymbol{r}_j = \boldsymbol{a} - \boldsymbol{v}_j = A/n - a_j$. Then, for any feasible $|\mathcal{J}| = k$,

$$\sum_{j\in\mathcal{J}} p_j \boldsymbol{r}_j = \frac{1}{n}\sum_{j\in\mathcal{J}} (A/n - a_j) = \frac{1}{n}\left(\frac{kA}{n} - \sum_{j\in\mathcal{J}} a_j\right) = \frac{1}{n}\left(\frac{A}{2} - \sum_{j\in\mathcal{J}} a_j\right).$$

This term vanishes if and only if $\sum_{j\in\mathcal{J}} a_j = A/2$, i.e., if and only if the balanced partition exists. Moreover, $P_{\mathcal{J}} = k/n = 1/2$ is constant over all feasible $\mathcal{J}$, so the denominator $1 - P_{\mathcal{J}}$ does not change the decision. Thus deciding whether $\min_{|\mathcal{J}|=k} F(\mathcal{J}) = 0$ is at least as hard as **Balanced Partition**, which is NP-complete (Hartmanis, 1982). The construction also satisfies Assumption 1: taking $\varepsilon = 1/n$ gives $k\varepsilon = 1/2 < 1$.

### C.2. Main Proofs

**Lemma 1** (Perturbation)**.** *Suppose a set of tokens indexed by $\mathcal{J} \subset [n]$ is evicted from the full KV cache, then the perturbation to the original attention output $\boldsymbol{a}$ associated with the query $\boldsymbol{q}$ is*

$$\Delta(\mathcal{J}) = \frac{\sum_{i\in\mathcal{J}} p_i(\boldsymbol{a} - \boldsymbol{v}_i)}{1 - P_{\mathcal{J}}},$$

*where $P_{\mathcal{J}} := \sum_{i\in\mathcal{J}} p_i$ denotes the total attention weights associated with the evicted tokens. In particular, for $\mathcal{J} = \{j\}$, we have the perturbation $\Delta(\{j\}) = \frac{p_j}{1-p_j}(\boldsymbol{a} - \boldsymbol{v}_j)$ for per-token eviction.*

*Proof of Lemma 1.* Let $\boldsymbol{a}_{-\mathcal{J}}$ be the attention output computed after removing the tokens in $\mathcal{J}$ from the KV cache. Because the softmax probabilities must be re-normalized over the remaining tokens. Specifically, $\forall\, i \notin \mathcal{J}$, it is easy to show that the associated re-normalized probability is given by $p'_i = \frac{p_i}{\sum_{j\notin\mathcal{J}} p_j}$.

Therefore, this updated attention output is given by

$$\boldsymbol{a}_{-\mathcal{J}} = \sum_{i\notin\mathcal{J}} p'_i \boldsymbol{v}_i = \frac{\sum_{i\notin\mathcal{J}} p_i \boldsymbol{v}_i}{\sum_{j\notin\mathcal{J}} p_j} = \frac{\boldsymbol{a} - \sum_{i\in\mathcal{J}} p_i \boldsymbol{v}_i}{1 - \sum_{i\in\mathcal{J}} p_i} = \frac{\boldsymbol{a} - \sum_{i\in\mathcal{J}} p_i \boldsymbol{v}_i}{1 - P_{\mathcal{J}}},$$

where we used $\sum_{i=1}^n p_i = 1$ and $\boldsymbol{a} = \sum_{i=1}^n p_i \boldsymbol{v}_i$. Thus,

$$\Delta(\mathcal{J}) = \boldsymbol{a}_{-\mathcal{J}} - \boldsymbol{a} = \frac{\boldsymbol{a} - \sum_{i\in\mathcal{J}} p_i v_i}{1 - P_{\mathcal{J}}} - \boldsymbol{a} = \frac{P_{\mathcal{J}}\boldsymbol{a} - \sum_{i\in\mathcal{J}} p_i \boldsymbol{v}_i}{1 - P_{\mathcal{J}}} = \frac{\sum_{i\in\mathcal{J}} p_i(\boldsymbol{a} - \boldsymbol{v}_i)}{1 - P_{\mathcal{J}}}.$$

□

**Theorem 1** (Near-Optimality). *Let $\mathcal{J}^* = \arg\min\{F(\mathcal{J}) := \|\Delta(\mathcal{J})\|_2 : \mathcal{J} \subseteq \mathcal{L},\ |\mathcal{J}| = k\}$ be the optimal eviction set and $\hat{\mathcal{J}} = \arg\operatorname{topk}^-_{j\in\mathcal{L}}(\|\Delta(\{j\})\|_2, k)$ the proxy. Under Assumptions 1 and 2, we have*

$$F(\hat{\mathcal{J}}) \le \frac{1}{c(1-\varepsilon)(1-k\varepsilon)}\, F(\mathcal{J}^*).$$

*Proof of Theorem 1.* For any $\mathcal{J} \subseteq \mathcal{L}$ with $|\mathcal{J}| = k$, we first define an auxiliary objective function:

$$G(\mathcal{J}) := \sum_{j\in\mathcal{J}} \|\Delta(\{j\})\|_2 = \sum_{j\in\mathcal{J}} \frac{p_j}{1-p_j}\|\boldsymbol{r}_j\|_2.$$

Then, by the definition of $\hat{\mathcal{J}} := \arg\operatorname{topk}^-_{j\in\mathcal{L}}(\|\Delta(\{j\})\|_2, k)$, it is easy to see that $G(\mathcal{J}) \ge G(\hat{\mathcal{J}}), \forall\, \mathcal{J} \subseteq \mathcal{L}$ with $|\mathcal{J}| = k$. That is, $\hat{\mathcal{J}}$ minimizes $G$. Moreover, using the triangle inequality, we have

$$G(\mathcal{J}) = \sum_{j\in\mathcal{J}} \frac{p_j}{1-p_j}\|\boldsymbol{r}_j\|_2 \ge \sum_{j\in\mathcal{J}} p_j\|\boldsymbol{r}_j\|_2 \ge \Big\|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\Big\|_2.$$

Hence, by Assumption 1, $P_{\mathcal{J}} \le k\varepsilon$, we have

$$F(\mathcal{J}) = \frac{\big\|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\big\|_2}{1-P_{\mathcal{J}}} \le \frac{G(\mathcal{J})}{1-P_{\mathcal{J}}} \le \frac{G(\mathcal{J})}{1-k\varepsilon}. \tag{2}$$

On the other hand, since $\boldsymbol{u}^\top\boldsymbol{z} \le \|\boldsymbol{u}\|_2\|\boldsymbol{z}\|_2 = \|\boldsymbol{z}\|_2$ for all $\boldsymbol{z} \in \mathbb{R}^d$, and by Assumption 2, $\boldsymbol{u}^\top\boldsymbol{r}_j \ge c\|\boldsymbol{r}_j\|_2$ for all $j \in \mathcal{J}$, we have

$$\Big\|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\Big\|_2 \ge \boldsymbol{u}^\top\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j \ge c\sum_{j\in\mathcal{J}} p_j\|\boldsymbol{r}_j\|_2.$$

Then, since $p_j \le \varepsilon$, we have

$$G(\mathcal{J}) = \sum_{j\in\mathcal{J}} \frac{p_j}{1-p_j}\|\boldsymbol{r}_j\|_2 \le \frac{1}{1-\varepsilon}\sum_{j\in\mathcal{J}} p_j\|\boldsymbol{r}_j\|_2 \le \frac{1}{c(1-\varepsilon)}\Big\|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\Big\|_2.$$

Therefore,

$$F(\mathcal{J}) = \frac{\big\|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\big\|_2}{1-P_{\mathcal{J}}} \ge \Big\|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\Big\|_2 \ge c(1-\varepsilon)G(\mathcal{J}). \tag{3}$$

Note that $G(\hat{\mathcal{J}}) \le G(\mathcal{J}^*)$, we have

$$F(\mathcal{J}^*) \ge c(1-\varepsilon)G(\mathcal{J}^*) \ge c(1-\varepsilon)G(\hat{\mathcal{J}}) \ge c(1-\varepsilon)(1-k\varepsilon)F(\hat{\mathcal{J}}),$$

where we used (3) in the first inequality and (2) in the last one. This completes the proof. □

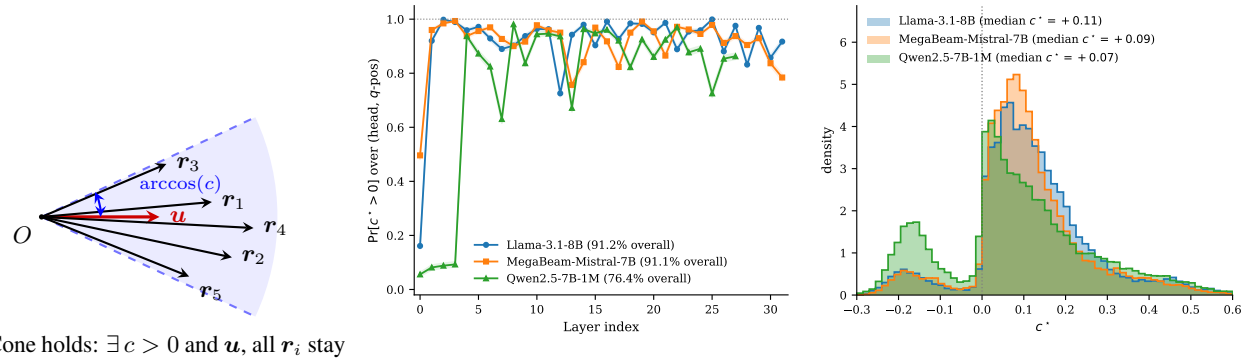
### C.3. Numerical Verification of Theorem Assumptions

This appendix reports two complementary empirical analyses of Theorem 1: (i) verification of the two assumptions and the resulting worst-case bound factor on real $n$=128K prefills, and (ii) the proxy's realized approximation ratio measured by brute-force enumeration on smaller-scale ($n$=2K) prefills, where the full per-query attention vector fits in memory under the eager backend. Both analyses cover Llama-3.1-8B-Instruct, MegaBeam-Mistral-7B-512K, and Qwen2.5-7B-Instruct-1M; for each model, we sample (layer, head, query-position) triples and extract the attention weights $\boldsymbol{p}$ and residuals $\boldsymbol{r}_j := \boldsymbol{a} - \boldsymbol{v}_j$. The candidate pool $\mathcal{L}$ is the low-attention eviction pool, excluding the final observation-window tokens.

**Assumption 1: small eviction probabilities.** For each triple, let $\varepsilon = \max_{j\in\mathcal{L}} p_j$. Table 12 shows that the typical low-attention candidate pool is in the small-weight regime: the median $\varepsilon$ is 2.6–4.3 $\times 10^{-6}$ and the 95th percentile is at most $\approx$1.2 $\times 10^{-4}$ across all three models. The maximum values are larger because a small number of early-layer/head triples contain isolated high-probability tokens; therefore we also report the finite-bound coverage below instead of claiming the union-bound denominator $1 - k\varepsilon$ is positive uniformly over all triples.

*Table 12.* Verification of Assumption 1 at $n = 128$K: distribution of $\varepsilon = \max_{j \in \mathcal{L}} p_j$ across sampled triples.

| Model | triples | median $\varepsilon$ | p95 $\varepsilon$ | max $\varepsilon$ | $\varepsilon \leq 0.005$ |
|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | 32K | $3.1\times10^{-6}$ | $1.1\times10^{-4}$ | 0.146 | 97.89% |
| MegaBeam-Mistral-7B-512K | 32K | $2.6\times10^{-6}$ | $9.7\times10^{-5}$ | 0.222 | 97.19% |
| Qwen2.5-7B-Instruct-1M | 25,088 | $4.3\times10^{-6}$ | $1.2\times10^{-4}$ | 0.176 | 97.52% |




*(a)* Cone holds: $\exists\, c > 0$ and $\boldsymbol{u}$, all $\boldsymbol{r}_i$ stay within $\arccos(c)$ of $\boldsymbol{u}$.

*(b)* Per-layer prevalence $\Pr[c^\star > 0]$.

*(c)* Distribution of $c^\star$.

*Figure 3.* Cone condition: definition and empirical verification at $n$=128K on three long-context models. **(a)** Residuals $\boldsymbol{r}_i := \boldsymbol{a} - \boldsymbol{v}_i$ inside a cone of half-angle $\arccos(c)$ about a single direction $\boldsymbol{u}$, so no positive combination cancels to zero. **(b)** Layerwise prevalence of $c^\star > 0$ across (head, query-position) triples, with binomial standard-error bands. **(c)** Distribution of $c^\star$ per model; mass to the right of the dotted line at $c^\star$=0 is the cone-holds region.

**Assumption 2: cone condition.** We verify Assumption 2 on real prefill passes at the RULER deployment length $n = 128$K across our three target long-context models: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), MegaBeam-Mistral-7B-512K (Wu & Song, 2025), and Qwen2.5-7B-Instruct-1M (Yang et al., 2025a). For each (layer, head, query-position) triple, we compute the cone constant:

$$c^\star \;:=\; \max_{\|\boldsymbol{u}\|=1} \min_{i \in \mathcal{L}} \frac{\boldsymbol{u}^\top \boldsymbol{r}_i}{\|\boldsymbol{r}_i\|_2},$$

where $\mathcal{L}$ is the candidate eviction pool; Assumption 2 holds at a given triple iff $c^\star > 0$. As shown in Fig. 3 (b)&(c), $c^\star > 0$ on 91.2%, 91.1%, and 76.5% of triples for Llama, Mistral, and Qwen2.5, respectively, with median $c^\star = +0.107, +0.094, +0.074$. Imposing both preconditions of Theorem 1 simultaneously ($c^\star > 0$ *and* $1 - k\varepsilon > 0$) leaves a finite multiplicative factor on 89.1%, 87.5%, and 73.4% of triples (Table 15); conditional on these preconditions, the realized factor $1/[c(1-\varepsilon)(1-k\varepsilon)]$ has median 10.4, 11.8, and 11.3 across the three models (full distribution in the same table). A direct brute-force study on small subsets of $\mathcal{L}$ shows the empirical approximation ratio is in fact much smaller than this worst-case bound (median $F(\hat{\mathcal{J}})/F(\mathcal{J}^*) \leq 1.16$ and 95th percentile $\leq 1.43$ across all three models; Appendix C.3, Table 16). Per-layer prevalence stays high from Layer 1 onward on Llama and Mistral; Llama Layer 0 ($\approx 16\%$) and Qwen2.5 Layers 0–3 (below 10%) exhibit pronounced early-layer dips (Fig. 3 b).

Table 13 reports the distribution of $c^\star$, and Table 14 reports the fraction of triples exceeding several positive thresholds. The strict cone condition holds on 91.21%, 91.12%, and 76.45% of triples for Llama, Mistral, and Qwen2.5, respectively.

*Table 13.* Cone constant $c^\star$ distribution across sampled triples at $n = 128$K.

| Model | min | p05 | **p50** | mean | p95 | max |
|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | −0.325 | −0.148 | **+0.107** | +0.120 | +0.389 | +0.882 |
| MegaBeam-Mistral-7B-512K | −0.334 | −0.131 | **+0.094** | +0.110 | +0.368 | +0.849 |
| Qwen2.5-7B-Instruct-1M | −0.567 | −0.196 | **+0.074** | +0.088 | +0.423 | +0.845 |

**Worst-case bound factor.** Combining the measured $(c^\star, \varepsilon)$ values with the union-bound denominator $1 - k\varepsilon$ from Theorem 1 yields the conservative multiplicative factor in Table 15. The factor is finite only when $c^\star > 0$ *and* $1 - k\varepsilon > 0$; over this finite subset, the median is 10.4, 11.8, and 11.3 across the three models, with heavy right tails (p95 between 115 and 198).

*Table 14.* Fraction of sampled triples satisfying $c^\star > \tau$.

| Model | $\tau = 0.00$ | 0.05 | 0.10 | 0.20 | 0.30 | 0.50 |
|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | 91.21% | 74.16% | 53.01% | 21.43% | 8.93% | 1.29% |
| MegaBeam-Mistral-7B-512K | 91.12% | 72.37% | 46.85% | 17.64% | 8.49% | 0.99% |
| Qwen2.5-7B-Instruct-1M | 76.45% | 57.29% | 42.99% | 22.50% | 12.33% | 2.14% |

*Table 15.* Empirical distribution of the conservative multiplicative factor in Theorem 1, computed as $1/[c^\star(1-\varepsilon)(1-k\varepsilon)]$ from the measured assumption constants. "finite" is the fraction of sampled triples with $c^\star > 0$ and $1 - k\varepsilon > 0$; p50/p95/max are computed over this finite subset.

| Model | finite | p50 | p95 | max |
|---|---|---|---|---|
| Llama-3.1-8B-Instruct | 89.13% | 10.42 | 122 | $8.2\times10^4$ |
| MegaBeam-Mistral-7B-512K | 87.53% | 11.83 | 115 | $1.5\times10^6$ |
| Qwen2.5-7B-Instruct-1M | 73.43% | 11.32 | 198 | $2.2\times10^5$ |

**Realized approximation ratio.** Beyond the worst-case bound characterized above, we directly measure the proxy's actual approximation ratio. For each sampled (layer, head, query-position) triple, we draw $n_{\text{small}}$=20 keys from the candidate pool $\mathcal{L}$ and brute-force enumerate all $\binom{20}{k}$ subsets ($k \in \{10, 18\}$) to find the joint optimum $F(\mathcal{J}^*) := \min_{\mathcal{J},|\mathcal{J}|=k} \| \sum_{i\in\mathcal{J}} p_i(\boldsymbol{a} - \boldsymbol{v}_i)\|_2/(1 - P_{\mathcal{J}})$. Four stratifications target progressively harder regimes: (i) uniform random sampling from $\mathcal{L}$; (ii) the *low-attention tail*, the bottom-$n_{\text{small}}$ entries by $p_j$; (iii) the *near-proxy-threshold* band, entries closest to the median DropKV score within $\mathcal{L}$, where the proxy is most likely to misrank; and (iv) the *rank-disagreement* band, entries where the attention-only and DropKV rankings diverge most, the regime where the value-residual factor matters most.

Across 1,200 records per model on Llama-3.1-8B-Instruct, MegaBeam-Mistral-7B-512K, and Qwen2.5-7B-Instruct-1M, the DropKV proxy ratio $F(\hat{\mathcal{J}})/F(\mathcal{J}^*)$ has median $\leq 1.16$ and 95th percentile $\leq 1.43$ in every (model $\times$ stratification $\times$ $k$) cell, with the worst single record at 1.82 (Table 16). Compared with the median bound factor of 10–12 in Table 15, the empirical realized ratio is roughly an order of magnitude tighter; the looseness of Theorem 1's constant therefore reflects worst-case mathematical conservatism rather than slack in the algorithm itself. DropKV and raw-attention selection behave nearly identically on uniform-random and low-attention-tail strata (medians within 0.02), but diverge on the rank-disagreement stratification (DropKV median 1.03 vs. attention-only median 1.09–1.15; 95th-percentile 1.17–1.26 vs. 1.49–1.58 across the three models), confirming that the value-residual factor $\|\boldsymbol{a} - \boldsymbol{v}_j\|_2$ is decisive precisely where attention rank alone is ambiguous. Random selection is consistently dominated, with median 1.16–2.93 at $k$=10.

*Table 16.* Empirical approximation gap: DropKV proxy ratio $F(\hat{\mathcal{J}})/F(\mathcal{J}^*)$ at $n_{\text{small}}$=20, broken down by stratification and $k$. Median (p50) and 95th percentile (p95) computed over 150 triples per (model, stratification, $k$) cell. Compare against the worst-case bound (median 10–12) in Table 15.

| | Llama-3.1-8B | | Mistral-7B-512K | | Qwen2.5-7B-1M | |
|---|---|---|---|---|---|---|
| Stratification | p50 | p95 | p50 | p95 | p50 | p95 |
| $k = 10$ (50% budget) | | | | | | |
| random | 1.000 | 1.095 | 1.000 | 1.107 | 1.000 | 1.107 |
| low-attention tail | 1.036 | 1.164 | 1.029 | 1.130 | 1.035 | 1.136 |
| near-threshold | 1.140 | 1.278 | 1.151 | 1.344 | 1.160 | 1.430 |
| rank-disagreement | 1.034 | 1.169 | 1.029 | 1.185 | 1.034 | 1.262 |
| $k = 18$ (10% budget, matches the deployment budget in §5) | | | | | | |
| random | 1.000 | 1.099 | 1.000 | 1.093 | 1.000 | 1.081 |
| low-attention tail | 1.012 | 1.066 | 1.012 | 1.079 | 1.011 | 1.073 |
| near-threshold | 1.032 | 1.084 | 1.032 | 1.109 | 1.038 | 1.109 |
| rank-disagreement | 1.007 | 1.104 | 1.000 | 1.085 | 1.003 | 1.094 |

The verification above is limited to Assumptions 1 and 2, which are the two assumptions used by Theorem 1. Assumption 3 below is a relaxed condition for the additive robustness result; verifying it directly would require estimating the out-of-cone weighted mass parameter $\mu$ for each triple, which we leave outside this numerical check.

### C.4. Further Robustness Analysis

When the candidate pool $\mathcal{L}$ does not strictly satisfy the cone condition as in Assumption 2, we establish near-optimality bound up to an additive error to account for the outlier value residuals $\boldsymbol{r}_i := \boldsymbol{a} - \boldsymbol{v}_i$. Specifically, we make the following relaxed assumption instead:

**Assumption 3** (Approximate Cone Condition)**.** *There exist constants $\mu \geq 0$, $c \in (0, 1]$, and a unit vector $\boldsymbol{u} \in \mathbb{R}^d$, such that*

$$\sum_{j \in \mathcal{L} \setminus \mathcal{L}_{\boldsymbol{u},c}} p_j \|\boldsymbol{r}_j\|_2 \leq \mu,$$

*where $\mathcal{L}_{\boldsymbol{u},c} := \{l \in \mathcal{L} : \boldsymbol{u}^\top \boldsymbol{r}_l \geq c\|\boldsymbol{r}_l\|_2\}$ denotes the set of indices whose residuals $\boldsymbol{r}_l$ reside within a primary alignment cone centered at $\boldsymbol{u}$.*

This relaxed assumption imposes that the weighted mass of outlier residuals is upper-bounded by $\mu$. Therefore, the strict cone condition, Assumption 2, is a special case of Assumption 3 with $\mathcal{L}_{\boldsymbol{u},c} = \mathcal{L}$ and $\mu = 0$; See Fig. 4 for an illustration.

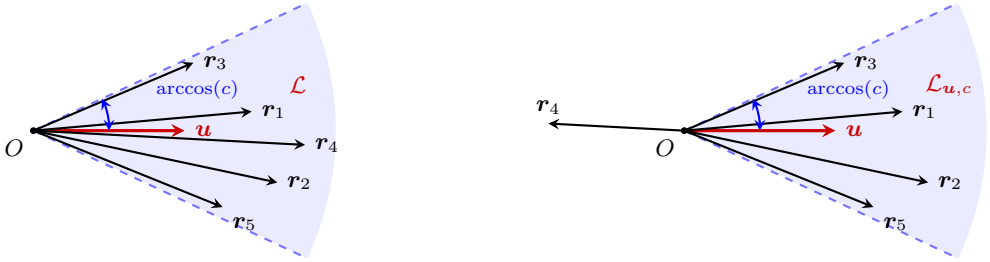



*Figure 4.* **Left:** Strict Cone Condition (Assumption 2). **Right:** Approximate Cone Condition (Assumption 3): $\mathcal{L}_{\boldsymbol{u},c} = \{1, 2, 3, 5\}$ and $p_4\|\boldsymbol{r}_4\|_2 \leq \mu$.

**Theorem 2** (Robust Near-Optimality)**.** *Let $\mathcal{J}^* = \arg\min\{F(\mathcal{J}) := \|\Delta(\mathcal{J})\|_2 : \mathcal{J} \subseteq \mathcal{L},\ |\mathcal{J}| = k\}$ be the optimal eviction set and $\hat{\mathcal{J}} = \arg\operatorname{topk}^-_{j \in \mathcal{L}}(\|\Delta(\{j\})\|_2, k)$ the proxy. Under Assumptions 1 and 3, we have*

$$F(\hat{\mathcal{J}}) \ \leq \ \frac{1}{c(1-\varepsilon)(1-k\varepsilon)} \Big( F(\mathcal{J}^\star) + (1+c)\mu \Big).$$

*Proof.* For any $\mathcal{J} \subseteq \mathcal{L}$ with $|\mathcal{J}| = k$, following the proof of Theorem 1, we define

$$G(\mathcal{J}) := \sum_{j \in \mathcal{J}} \|\Delta(\{j\})\|_2 = \sum_{j \in \mathcal{J}} \frac{p_j}{1-p_j} \|\boldsymbol{r}_j\|_2.$$

Then, $G(\mathcal{J}) \geq G(\hat{\mathcal{J}}), \forall\, \mathcal{J} \subseteq \mathcal{L}$ with $|\mathcal{J}| = k$. Moreover, we have $F(\mathcal{J}) \ \leq \ \frac{G(\mathcal{J})}{1-k\varepsilon}$.

On the other hand, we have

$$\begin{aligned} \Big\| \sum_{j \in \mathcal{J}} p_j \boldsymbol{r}_j \Big\|_2 \geq \boldsymbol{u}^\top \sum_{j \in \mathcal{J}} p_j \boldsymbol{r}_j &= \boldsymbol{u}^\top \sum_{j \in \mathcal{J} \cap \mathcal{L}_{\boldsymbol{u},c}} p_j \boldsymbol{r}_j + \boldsymbol{u}^\top \sum_{j \in \mathcal{J} \setminus \mathcal{L}_{\boldsymbol{u},c}} p_j \boldsymbol{r}_j \\ &\geq c \sum_{j \in \mathcal{J} \cap \mathcal{L}_{\boldsymbol{u},c}} p_j \|\boldsymbol{r}_j\|_2 - \sum_{j \in \mathcal{J} \setminus \mathcal{L}_{\boldsymbol{u},c}} p_j \|\boldsymbol{r}_j\|_2, \end{aligned}$$

using $\boldsymbol{u}^\top \boldsymbol{r}_j \geq c\|\boldsymbol{r}_j\|_2$ on $\mathcal{L}_{\boldsymbol{u},c}$ and $\|\boldsymbol{z}\|_2 \geq \boldsymbol{u}^\top \boldsymbol{z} \geq -\|\boldsymbol{z}\|_2$ for all $\boldsymbol{z} \in \mathbb{R}^d$.

Rearranging and adding the term $c \sum_{j \in \mathcal{J} \setminus \mathcal{L}_{\boldsymbol{u},c}} p_j \|\boldsymbol{r}_j\|_2$ on both sides gives

$$c \sum_{j \in \mathcal{J}} p_j \|\boldsymbol{r}_j\|_2 \leq \Big\| \sum_{j \in \mathcal{J}} p_j \boldsymbol{r}_j \Big\|_2 + (1+c) \sum_{j \in \mathcal{J} \setminus \mathcal{L}_{\boldsymbol{u},c}} p_j \|\boldsymbol{r}_j\|_2.$$

Since $p_j \leq \varepsilon$ implies $G(\mathcal{J}) \leq \frac{1}{1-\varepsilon} \sum_{j \in \mathcal{J}} p_j \|\boldsymbol{r}_j\|_2$, we obtain

$$G(\mathcal{J}) \leq \frac{1}{c(1-\varepsilon)} \Big( \Big\| \sum_{j \in \mathcal{J}} p_j \boldsymbol{r}_j \Big\|_2 + (1+c) \sum_{j \in \mathcal{J} \setminus \mathcal{L}_{\boldsymbol{u},c}} p_j \|\boldsymbol{r}_j\|_2 \Big)$$

$$\leq \frac{1}{c(1-\varepsilon)}\Big(F(\mathcal{J}) + (1+c)\sum_{j\in\mathcal{J}\setminus\mathcal{L}_{\boldsymbol{u},c}} p_j\|\boldsymbol{r}_j\|_2\Big)$$
$$\leq \frac{1}{c(1-\varepsilon)}\Big(F(\mathcal{J}) + (1+c)\mu\Big) \tag{4}$$

where we used $F(\mathcal{J}) = \|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\|_2/(1-P_{\mathcal{J}}) \geq \|\sum_{j\in\mathcal{J}} p_j\boldsymbol{r}_j\|_2$ and Assumption 3:

$$\sum_{j\in\mathcal{J}\setminus\mathcal{L}_{\boldsymbol{u},c}} p_j\|\boldsymbol{r}_j\|_2 \leq \sum_{j\in\mathcal{L}\setminus\mathcal{L}_{\boldsymbol{u},c}} p_j\|\boldsymbol{r}_j\|_2 \leq \mu.$$

Applying (4) to $\mathcal{J} = \mathcal{J}^\star$, we have

$$G(\mathcal{J}^*) \leq \frac{1}{c(1-\varepsilon)}\Big(F(\mathcal{J}^*) + (1+c)\mu\Big)$$

Finally, the facts $G(\widehat{\mathcal{J}}) \leq G(\mathcal{J}^\star)$ and $F(\widehat{\mathcal{J}}) \leq \frac{G(\widehat{\mathcal{J}})}{1-k\varepsilon}$ yield the claim. □

## D. Triton Implementation Details

### D.1. Triton Kernel Implementation

A naïve PyTorch realization of Algorithm 1 materializes dense window-by-cache tensors per head: the attention weights have shape $w \times n$, and a direct broadcasted implementation of the value-residual term forms $w \times n \times d$ intermediates. This memory and bandwidth pattern grows as $O(wnd)$ and dominates the scoring path at long context. We replace it with three fused Triton kernels — a FlashAttention-2-style forward pass, the scoring kernel, and a split-K merge — that compute $\mathrm{score}_j$ in (DropKV) by streaming over key/value tiles without materializing the attention-weight or value-distance tensors.

The scoring kernel is the central contribution. Using the row-wise Log-Sum-Exp $\mathrm{LSE}^{(t)} := \log\sum_{j=1}^{n}\exp(\boldsymbol{K}\boldsymbol{q}^{(t)}/\sqrt{d})_j$ produced by the forward pass (Dao, 2023), it streams over key tiles, recomputes logits $\mathrm{logit}_j = \boldsymbol{k}_j^\top\boldsymbol{q}^{(t)}/\sqrt{d}$ on-the-fly, derives the attention weight $p_j^{(t)} = \exp(\mathrm{logit}_j - \mathrm{LSE}^{(t)})$, and accumulates $\mathrm{score}_j \mathrel{+}= \big(p_j^{(t)}/(1-p_j^{(t)})\big)^2\|\boldsymbol{a}^{(t)} - \boldsymbol{v}_j\|_2^2$ across query indices $t$. By precomputing $\|\boldsymbol{a}^{(t)}\|_2^2$ and $\|\boldsymbol{v}_j\|_2^2$ once and using the identity $\|\boldsymbol{a}-\boldsymbol{v}\|_2^2 = \|\boldsymbol{a}\|_2^2 + \|\boldsymbol{v}\|_2^2 - 2\langle\boldsymbol{a},\boldsymbol{v}\rangle$, each $\boldsymbol{V}$ tile is read only once per pass, substantially reducing I/O.

The auxiliary forward and split-K merge kernels follow standard FlashAttention-style online softmax; detailed pseudocode for the scoring kernel is in Appendix D. The fused path matches the PyTorch reference to within `BF16` round-off, with a scoring-step speedup over the naïve PyTorch baseline that grows from $1.8\times$ at $n = 4$K to $19.8\times$ at $n = 128$K ($w = 8$, A100-80GB; see Fig. 2 left).

**Algorithm 1** DropKV: Decoupled Residual-Output Perturbation for KV-Cache Eviction.

**Require:** keys $\boldsymbol{K} \in \mathbb{R}^{n\times d}$, values $\boldsymbol{V} \in \mathbb{R}^{n\times d}$, query window $\mathcal{Q} = \{\boldsymbol{q}^{(t)}\}_{t=1}^{w}$, eviction size $k$, pooling kernel size $\kappa$
**Ensure:** Eviction set $\widehat{\mathcal{J}} \subseteq [n]$ with $|\widehat{\mathcal{J}}| = k$
$\forall\, t \in [w]:\ \boldsymbol{p}^{(t)} \leftarrow \mathrm{softmax}(\boldsymbol{K}\boldsymbol{q}^{(t)}/\sqrt{d}),\ \boldsymbol{a}^{(t)} \leftarrow \boldsymbol{V}^\top\boldsymbol{p}^{(t)}$
**for** $j = 1, \ldots, n$ **do**
  $\mathrm{score}_j \leftarrow \sum_{t=1}^{w}\big(p_j^{(t)}/(1-p_j^{(t)})\big)^2\|\boldsymbol{a}^{(t)} - \boldsymbol{v}_j\|_2^2$
**end for**
$\mathrm{score} \leftarrow \mathrm{MaxPool1D}(\mathrm{score},\ \kappa);\ \mathrm{score}_j \leftarrow +\infty$ for $j \in \{n-w+1, \ldots, n\}$
$\widehat{\mathcal{J}} \leftarrow$ indices of the $k$ smallest $\mathrm{score}_j$
**return** $\widehat{\mathcal{J}}$

This section describes the Triton implementation sketched in §D.1. For clarity, the notation is written for one attention head; the actual kernels are launched independently over the batch-head axis. The notation is head-local; model-specific head layout is handled before this scoring interface. Queries have shape $\boldsymbol{Q} \in \mathbb{R}^{B\times H\times W\times D}$, keys and values have shape $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{B\times H\times L\times D}$, and the output score tensor has shape $\mathrm{score} \in \mathbb{R}^{B\times H\times L}$. Let $s := 1/\sqrt{D}$, let $B_n$ denote the key/value tile size, and let $B_k$ denote the inner dimension tile used for dot products.

**Algorithm 2** Kernel B: DropKV score accumulation

**Require:** $\boldsymbol{q}^{(1:W)} \in \mathbb{R}^{W\times D}$, $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{L\times D}$, outputs $\boldsymbol{a}^{(1:W)}$, Log-Sum-Exp $L^{(1:W)}$
**Require:** key tile $\mathcal{T} = \{j_0, \dots, j_0 + B_n - 1\}$, scale $s$
**Ensure:** $\text{score}_j$ for $j \in \mathcal{T}$
Load $\boldsymbol{k}_j, \boldsymbol{v}_j$ for $j \in \mathcal{T}$; precompute $\|\boldsymbol{v}_j\|_2^2$
score $\leftarrow 0^{B_n}$
**for** $t = 1, \dots, W$ **do**
  $\text{logits}_j \leftarrow s\, \boldsymbol{k}_j^\top \boldsymbol{q}^{(t)}$ for all $j \in \mathcal{T}$; apply causal and padding masks
  $p_j \leftarrow \exp(\text{logits}_j - L^{(t)})$
  $\tilde{p}_j \leftarrow \text{cast}_{\texttt{BF16}}(p_j)$ {matches the `BF16` reference path}
  $\omega_j \leftarrow \left(\tilde{p}_j/(1 - \tilde{p}_j + \epsilon)\right)^2$
  $d_j^2 \leftarrow \|\boldsymbol{a}^{(t)}\|_2^2 + \|\boldsymbol{v}_j\|_2^2 - 2\langle \boldsymbol{a}^{(t)}, \boldsymbol{v}_j\rangle$
  $\text{score}_j \leftarrow \text{score}_j + \omega_j d_j^2$ for all $j \in \mathcal{T}$
**end for**
Write $\text{score}_j$ for $j \in \mathcal{T}$

**Auxiliary forward and split-K merge.** Kernel A is a standard FlashAttention-style forward pass over the $W$ window queries. For each query $\boldsymbol{q}^{(t)}$, it computes the attention output $\boldsymbol{a}^{(t)}$ and retains the row-wise log-sum-exp

$$L^{(t)} = \log \sum_{j=1}^{L} \exp\left(s\, \boldsymbol{k}_j^\top \boldsymbol{q}^{(t)}\right),$$

which allows Kernel B to recover $p_j^{(t)}$ without materializing the attention matrix. When split-K is enabled, Kernel A writes partial online-softmax states $(m_r, \ell_r, \text{acc}_{y,r})$ for split $r$, and the standard merge Kernel C computes

$$m^\star = \max_r m_r, \qquad \ell^\star = \sum_r e^{m_r - m^\star} \ell_r, \qquad \text{acc}_y^\star = \sum_r e^{m_r - m^\star} \text{acc}_{y,r},$$

then returns $\boldsymbol{a} = \text{acc}_y^\star/\ell^\star$ and $L = m^\star + \log \ell^\star$. These two auxiliary kernels are unchanged from the usual online-softmax computation except that $L^{(t)}$ is kept for the scoring pass.

**Launch and numerical guards.** Kernel B is launched over $(B \cdot H, \lceil L/B_n \rceil)$, so each program owns one contiguous key tile for one batch-head pair. Causal and padding masks are applied to logits before either the log-sum-exp or the score is formed; padded query rows may be zero-filled only after being excluded by the mask. The $\epsilon$ in $(1 - p + \epsilon)$ avoids division near $p \to 1$. The `BF16` cast of $p$ before squaring is used to match the `BF16` PyTorch reference path; without it, precision loss in $(1 - p)$ can saturate the denominator differently and flip top-$k$ decisions on peak-attention tokens. We verify the Triton and PyTorch paths with a precision-matching test suite.

### D.2. GPU Profiling of the Triton Scoring Kernels

We further profile the scoring path with Nsight Compute (`ncu`, `--set full`) on an A100-80GB GPU using the Llama-3.1-8B shape ($B$=1, $H_q$=32, $H_{kv}$=8, $D$=128, $W$=8) in `BF16`. The goal of this profiling is to explain the implementation advantage behind Fig. 2: DropKV's Triton scorer is not merely a faster implementation of the same dense computation, but removes the dominant materialized attention and repeated-KV traffic from the scoring path.

**Full-call advantage over PyTorch.** Table 17 compares the matched PyTorch scoring reference with the Triton scoring pipeline at $n$=16K. Triton is 15.7× faster in this isolated `ncu` run, launches fewer kernels, executes 41× fewer instructions, and moves 26× less HBM traffic. The PyTorch reference has higher aggregate SM throughput and active-warps only because it creates far more work by materializing dense $W \times L$ attention tensors and repeated K/V tensors. In contrast, the Triton path streams K/V tiles and accumulates scores directly, which is the desired behavior for long-context KV scoring.

**Per-kernel behavior.** The Triton scoring call consists of three named kernels: Kernel A computes the auxiliary FlashAttention-style forward/log-sum-exp, Kernel B accumulates DropKV scores, and Kernel C merges split-K par-

*Table 17.* Nsight Compute full-call profiling at $n$=16K. Both columns aggregate all launches inside the scoring call: time/bytes/ instructions are summed, percentage metrics are time-weighted, and register or shared-memory footprints are maxima over launches.

| Metric | PyTorch scorer | Triton scorer |
|---|---|---|
| Runtime (ms) | 3.63 | **0.23** |
| Speedup | 1.0× | **15.7**× |
| Kernels launched | 34 | **8** |
| HBM read (MB) | 2,175 | **138** |
| HBM write (MB) | 1,545 | **5** |
| Mem% of peak HBM | 50.3 | 30.4 |
| SM throughput (%) | 48.4 | 18.1 |
| Active warps (%) | 77.3 | 12.8 |
| HMMA pipe (%) | 4.6 | **10.1** |
| FMA pipe (%) | 27.4 | 15.3 |
| Instructions (M) | 851 | **20.5** |
| Regs/thread (max) | 150 | 154 |
| SMEM/block, KB (max) | 99.3 | 37.9 |

*Table 18.* Per-kernel `ncu` measurements for the Triton scoring pipeline at $n$=16K. SM% and Mem% are fractions of peak sustained SM issue and peak HBM bandwidth. Active warps is measured achieved occupancy; Ceiling is the resource-limited occupancy ceiling.

| Kernel | Time ($\mu$s) | SM% | Mem% | Active warps | Ceiling | Main limiter |
|---|---|---|---|---|---|---|
| A: forward/LSE | 69.82 | 22.7 | **49.0** | 11.6% | 18.75% | Registers |
| B: score accumulation | 63.52 | 39.6 | **54.0** | 22.6% | 25.00% | SMEM |
| C: split-K merge | 78.66 | 0.5 | 2.3 | 6.3% | 43.75% | Reduction overhead |

tial states. The named A/B/C launches in Table 18 sum to 212.0 $\mu$s, while the aggregate Triton row in Table 17 is 231.5 $\mu$s over all eight launches; the remaining auxiliary launches therefore add only 19.5 $\mu$s. This confirms that the two `ncu` views are internally consistent.

At long context, Kernels A and B are memory-throughput limited: at $n$=16K, Kernel A reaches 49.0% of peak HBM bandwidth versus 22.7% SM throughput, while Kernel B reaches 54.0% peak HBM bandwidth versus 39.6% SM throughput. Kernel B alone therefore sustains roughly 1.10 TB/s on A100 HBM, despite the low aggregate active-warps in Table 17. The low active-warps are not evidence of wasted dense computation; they reflect resource ceilings from registers and shared memory, while the useful long-context kernels are already limited primarily by streaming K/V traffic.

**Why this supports the three-kernel design.** The profiling identifies the remaining bottleneck as structural rather than an avoidable PyTorch-style materialization overhead. Kernel A uses 154 registers per thread and is capped at 12 resident warps/SM; Kernel B uses 37.9 KB shared memory per block and is capped at 16 resident warps/SM. A persistent fused A+B kernel would collapse Kernel B's parallel grid from $(B \cdot H_{kv}, \lceil L/B_n \rceil)$ to $(B \cdot H_{kv},)$ after the forward phase, leaving only eight CTAs for Llama-3.1-8B at $B$=1 and substantially reducing memory-level parallelism. Thus the current split keeps enough tile-level parallelism for the memory-bound scoring kernel, while still avoiding the PyTorch reference's dense attention matrix and repeated-KV tensors. This is the central implementation advantage of DropKV: it turns an otherwise materialization-heavy scoring formula into a compact, streaming, bandwidth-efficient Triton pipeline.

## E. Details of RULER and LongBench Benchmarks

**RULER:** Below is a brief overview of the different tasks in RULER.

- **Single NIAH (S-NIAH):** A basic information retrieval task. In this scenario, a keyword sentence, referred to as the "needle", is embedded within a lengthy text, called the "haystack". The goal is to retrieve the "needle" from the context. The "haystack" may consist of repetitive, noisy sentences or essays by Paul Graham (Kamradt., 2023).
- **Multi-keys NIAH (MK-NIAH):** Multiple "needles" are inserted into the "haystack", but the task requires retrieving only one of them.

- **Multi-values NIAH (MV-NIAH):** Multiple "needles" in the form of key-value pairs are placed within the "haystack". The task is to retrieve all values associated with a given key.
- **Multi-queries NIAH (MQ-NIAH):** Multiple "needle" in the form of key-value pairs are inserted into the "haystack". All "needles" corresponding to the keys specified in the queries need to be retrieved.
- **Variable Tracking (VT):** A series of variable binding statements (e.g., $X2 = X1, X3 = X2, ...$) are inserted at various positions within a long text to simulate the recognition of entities and relationships. The task objective is to return all variable names that point to the same value $V$.
- **Common words extraction task (CWE):** The input sequence is constructed by sampling words from a discrete uniform distribution within a pre-defined (synthetic) word list. Words that appear most frequently are termed "common words". The model is required to identify all common words. The number of common words remains constant while the number of uncommon words increases with the length of the sequence.
- **Frequent words extraction task (FWE):** The input sequence is constructed by sampling words from a Zeta distribution within a pre-defined (synthetic) word list. The model need to return the top-K frequent words in the context.
- **Single-Hop QA (S-QA)** Answer questions based on passages. The context is extended to simulate long-context input by adding distracting information(golden paragraphs). And answers come from a single piece of evidence.
- **Multi-Hop QA (M-QA)** Answer questions based on passages. For Multi-hop QA, which requires integrating information from multiple sources.

**LongBench** spans multiple task domains with an average length of 6711, including single-document QA (Dasigi et al., 2021), multi-document QA (Ho et al., 2020a; Trivedi et al., 2022; Yang et al., 2018), summarization (Fabbri et al., 2019; Huang et al., 2021; Zhong et al., 2021), few-shot learning (Gliwa et al., 2019; Joshi et al., 2017; Li & Roth, 2002), synthetic tasks (Bai et al., 2024), and code generation (Guo et al., 2023; Liu et al., 2023b).

- **Single-Doc QA:** Single-document QA requires models to obtain the answer from a single piece of source. The test samples are derived from diverse datasets including NarrativeQA, qasper (Dasigi et al., 2021), and MultiFieldQA (Liu et al., 2023a), encompassing a range of documents such as legal files, governmental reports, encyclopedia, and academic papers.
- **Multi-Doc QA:** Multi-document QA requires models to extract and combine information from several documents to obtain the answer. The test samples are built from three Wikipedia-based multi-hop QA datasets: HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020b) and MuSiQue (Trivedi et al., 2022).
- **Summarization:** Summarization task requires a comprehensive understanding of the context. The test samples are drawn from the GovReport dataset (Huang et al., 2021), and QMSum (Zhong et al., 2021). Additionally, the MultiNews dataset originates from a multi-document summarization corpus detailed in (Fabbri et al., 2019)
- **Few-shot Learning:** Few-shot Learning task have integrated classification, summarization, and reading comprehension tasks to maintain task diversity. For classification, the TREC dataset (Li & Roth, 2002) is included. The SAMSum dataset (Gliwa et al., 2019), which comprises messenger-style conversations with human-annotated summaries, has been incorporated for summarization tasks. Additionally, TriviaQA (Joshi et al., 2017), a dataset of question-answer pairs, is utilized for reading comprehension tasks.
- **Synthetic Task:** Synthetic Tasks are designed to assess the model's ability in handling specific scenarios and patterns. Two synthetic tasks, PassageRetrieval-en and PassageCount, are included in LongBench. PassageRetrieval-en is derived from English Wikipedia. 30 passages is randomly selected and use GPT-3.5-Turbo to summarize one of them. The task challenges the model to identify the original paragraph that matches the generated summary. PassageCount is designed to be more challenging, this task requires the model to leverage the entire context to solve it. Several passages are randomly pickedfrom English Wikipedia, randomly duplicate each paragraph multiple times, and then shuffle the paragraphs. The task is to determine the number of unique passages within the provided set.
- **Code Completion:** Code Completion task is to assist users by completing code based on previous code input and context. The LCC dataset, derived from the original Long Code Completion dataset (Guo et al., 2023). The RepoBench-P dataset (Liu et al., 2023b) is designed to aggregating information from code across files. Model is required to predict the next line of code.