KEYDIFF: Key Similarity-Based KV Cache Eviction for Long-Context LLM Inference in Resource-Constrained Environments

Junyoung Park[*] **Dalton Jones** **Matthew J Morse**
Raghavv Goel **Mingu Lee** **Chris Lott**
Qualcomm AI Research
{junpark,daltjone,mattmors,raghgoel,mingul,clott}@qti.qualcomm.com

Abstract

We demonstrate that geometrically distinctive keys during LLM inference tend to have high attention scores. Based on the phenomenon we propose KEYDIFF, a training-free KV cache eviction method based solely on key similarity. Unlike other KV cache eviction methods, KEYDIFF can process arbitrarily long prompts within strict resource constraints and efficiently generate responses. We provide a theoretical basis for KEYDIFF by relating key diversity with attention scores. These results imply KEYDIFF can efficiently identify the most important tokens to retain. Notably KEYDIFF does not rely on attention scores, allowing the use of optimized attention mechanisms like FlashAttention. Under a strict memory allowance, we demonstrate the effectiveness of KEYDIFF for the Llama and Qwen model families by observing a performance gap of less than 0.04% with 8K cache budget ($\sim$ 23% KV cache reduction) from the non-evicting baseline on LongBench for Llama 3.1-8B and Llama 3.2-3B. We also observe near baseline performance for Deepseek-R1-Distill-Llama-8B on the Math500 reasoning benchmark and decrease end-to-end inference latency by up to 30% compared to the other token-eviction methods.

1 Introduction

Key-Value (KV) caching is a standard technique to accelerate large language model (LLM) inference that reuses key and value states (KVs) from previously processed tokens, enabling efficient autoregressive generation. This is crucial for long-context applications such as document summarization, code generation, question answering [7, 25, 31, 10], retrieval augmented generation [19] and reasoning [34, 17, 41]. However, the memory footprint of the stored KV cache grows linearly with input length, which becomes a bottleneck in memory-constrained environments.

This challenge is particularly acute for LLM inference on edge device, where compute, memory, and power resources are limited [3, 22, 32, 36]. While *KV cache eviction* policies have been proposed to bound memory overhead by removing unimportant KVs (often measured by attention scores) [35, 24, 43], they typically process the entire prompt at once and violate memory constraints during intermediate computation.

To enforce strict memory bounds throughout the prompt prefill and token generation inference phases, we adopt a block-wise inference strategy: the input prompt is divided into smaller blocks which are processed sequentially by the model, similar to [18, 1, 37]. After processing each block, we evict some of the cached KVs by according to an eviction policy that scores each KV, as illustrated in Figure 1. Unlike previous approaches that apply eviction after processing the entire prompt, this

[*] Corresponding author

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

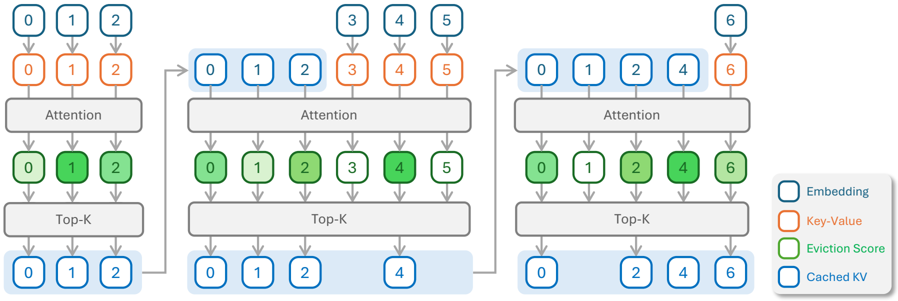


Figure 1: **An example of block prompt processing with KV cache eviction.** The input prompt having length of 7 is segmented by three blocks, and a transformer layer in LLM processes each block by **(1)** computing key-value states from inputs, **(2)** computing attention, **(3)** computing the eviction score, and **(4)** performing eviction based on the eviction score to satisfy the memory constraints (e.g., at most 4 tokens can reside in the cache). After each block processing, the KV cache is updated and passed to the next round of block processing, satisfying imposed memory constraints on the KV cache.

strategy satisfies memory constraints throughout the full inference process. However, we observe a degradation in accuracy when applying existing eviction methods in this setting (Table 1).

We hypothesize that the performance drop stems from a mismatch in design: existing eviction methods assume access to full-prompt attention, where key importance is computed over the entire input. During block prompt processing, however, attention is computed using only the current block's tokens without access to future blocks. As a result, attention scores based on a limited context often fail to reflect a token's true importance across the full prompt.

To this end, we observe that keys with lower average pairwise cosine similarity tend to receive higher attention scores across a variety of inputs. This suggests that key diversity serves as a strong proxy for global token importance, even without access to future tokens. This insight enables an attention-free approach to cache eviction that is based on the geometry of the cached keys.

Motivated by these observations, we propose KEYDIFF, an attention-free cache eviction method that removes redundancy among cached keys, operates effectively during block-wise inference, and avoids excessive memory overhead. Our contributions are summarized as follows:

- **Insight.** We observe that lower pairwise cosine similarity among keys correlates with higher attention scores, suggesting its utility as a proxy for token importance. (Section 3.1)
- **Method.** We introduce KEYDIFF, an eviction strategy that selects keys based on their similarity to other cached entries without relying on attention scores or future tokens. (Section 3.2)
- **Theory.** Through our analysis of key and query geometry we provide a theoretical understanding how/why KEYDIFF works. We also show that KEYDIFF solves an optimal subset selection problem that maximizes key diversity. (Section 3.3)
- **Performance.** KEYDIFF achieves $\leq 1.5\%$ and $\leq 0.04\%$ accuracy drop on LongBench with 6K and 8K cache budgets, respectively, outperforming state-of-the-art eviction methods across Llama and Qwen models (Section 4.2), and near non-evicting baseline performance for Deepseek-R1-Distill-Llama-8B on the Math-500 reasoning benchmark. (Section 4.3)
- **Efficiency.** We observe up to 30% end-to-end inference latency reduction using KEYDIFF compared to existing KV cache eviction methods. (Section 4.5)

2 Background

2.1 Transformers

The Transformer architecture [33] processes input data using a sequence of transformer blocks. A transformer block f takes a sequence $X = (x_1, x_2, \ldots, x_T) \in \mathbb{R}^{T\times d}$ as input and applies the causal self-attention operator $\mathrm{Attention}$ followed by a feed-forward network FF with optional gating [28] to produce the output $X' = (x'_1, x'_2, \ldots, x'_T) \in \mathbb{R}^{T\times d}$:

$$X' = f(X) = \mathrm{FF}(\mathrm{Attention}(X)), \tag{1}$$

The causal $\mathrm{Attention}$ operator projects each input token x_t with matrices $W_q, W_k, W_v \in \mathbb{R}^{d \times d}$ into key, query, and value matrices ($K = XW_k$, $Q = XW_Q$, $V = XW_V$, respectively) then applies the following relation to produce the attention output [2]:

$$O^{\text{attn}} = \text{Softmax}\left(QK^\top \big/ \sqrt{d} + M\right) V = AV \tag{2}$$

where $O^{\text{attn}} \in \mathbb{R}^{T \times d}$, and the causal attention mask M is an upper triangular matrix with nonzero values of $-\infty$.

2.2 KV Caching

When the $\mathrm{Attention}$ operator processes a new token x_{T+1}, it must also recompute the prior KV states for tokens $x_0, \ldots, x_T$. This can be avoided by storing previously computed KVs in a *KV cache* $\mathcal{C} = (K, V)$ for later reuse and append the new KV corresponding x_{T+1} to the cache. We can apply Equation (2) to an existing KV cache $\mathcal{C}$ as follows:

$$o^{\text{attn}}_{T+1} = \text{Softmax}\left(q_{T+1}[K \| k_{T+1}]^\top \big/ \sqrt{d} + M\right) [V \| v_{T+1}], \tag{3}$$

where $k_{T+1}, q_{T+1}, v_{T+1}$ are the key, query, and value states of x_{T+1}, and $[X \| x_{T+1}]$ represents the concatenation of x_{T+1} to an existing tensor X along the time dimension, M is the causal attention mask accounting for both the KV cache and x_{T+1}.

KV caching dramatically reduces the latency of $\mathrm{Attention}$ by only computing $k_{T+1}, q_{T+1}, v_{T+1}$ for each token x_{T+1} and reusing the KVs in $\mathcal{C}$. However, the size of the KV cache increases linearly with the number of processed tokens and dominates the memory footprint in long-context applications [39].

2.3 KV Cache Eviction Methods

To limit the memory footprint of the KV cache, we fix a *cache budget* N, which is the maximum number of tokens to be stored in the cache. If a new KV is added to the cache and the updated cache size is greater than N, we must evict KVs from the cache until the cache budget is met. The *eviction policy* $\pi_N(\mathcal{C})$ evicts a subset of KVs from $\mathcal{C}$ and returns a new cache $\mathcal{C}'$ containing at most N KVs:

$$\begin{aligned} \mathcal{C} &\leftarrow ([K \| k_{t+1}], [V \| v_{t+1}]) \\ \mathcal{C}' &\leftarrow \pi_N(\mathcal{C}) \end{aligned} \tag{4}$$

Attention-Based Eviction Policies Attention-based eviction policies π^{attn}_N use aggregated attention values to rank each KVs' relative importance and keep the N highest scoring KVs. For a given attention weight aggregation function ϕ, the eviction policy π^{attn}_N performs the following steps:

$$\begin{aligned} S &= \texttt{topk}(\phi(A), N) \\ K' &= \texttt{gather}(K, S), \quad V' = \texttt{gather}(V, S) \end{aligned} \tag{5}$$

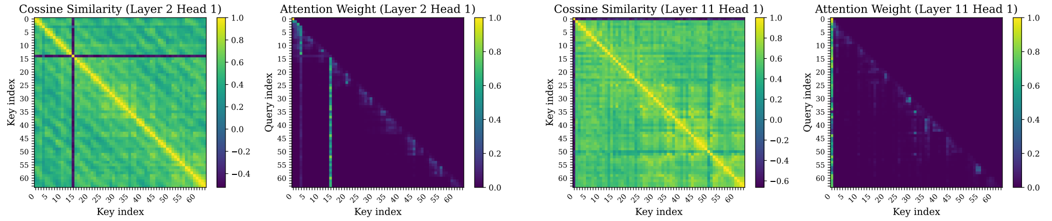
where $\texttt{topk}(x, N)$ returns the indices of N-largest values of x and $\texttt{gather}(X, S)$ gathers columns indexed by S.

Attention-based eviction methods prioritize KV pairs with higher attention scores to past tokens. This is problematic when applying block prompt processing: all input tokens are not simultaneously accessible within $\mathrm{Attention}$, only those in the current block and cache. This can result in an incorrect eviction decision. Additionally, attention-based eviction often requires explicitly materializing A, which can be resource intensive. We discuss the attention-based eviction policies further in Section 5.

2.4 KV Caching in Resource-Constrained Environments

Existing eviction policies like Zhang et al. [43], Oren et al. [24] focus on processing the entire input prompt *at once*: KVs are computed for each token in the prompt and stored in a cache $\mathcal{C}$, then the eviction policy π_N is applied to reduce the number of tokens in $\mathcal{C}'$ to N, before token generation.

[2] The multi-head extension and output projections are omitted for brevity.



(a) Layer 2 and Head 1

(b) Layer 11 and Head 1

Figure 2: **Cosine similarity of the keys and attention weights.** Measured from Llama 3.2-3B-Instruct and the first sample from the NarrativeQA dataset in LongBench. Truncated to the first 64 tokens for visualization.

However, the intermediate cache $\mathcal{C}$ before eviction will grow to the size of the input prompt. This can often exceed model's allocated memory limit when deploying long context applications in resource constrained environments.

As demonstrated in efficient LLM inference frameworks [1, 14, 18, 36], one solution is to apply π_N more frequently by segmenting X into non-overlapping blocks $X = [X_0, X_1, \ldots, X_{m-1}]$, where $X_i = [x_{Bi}, \ldots, x_{B(i+1)-1}]$, B is the block size, and $m = \lceil T/B \rceil$, and iteratively updating the cache by exploiting causality, applying Equation (4) in a block-wise fashion:

$$\begin{aligned} &\mathcal{C}_i \leftarrow ([K_{i-1} \| k_{Bi:B(i+1)-1}], [V_{i-1} \| v_{Bi:B(i+1)-1}]) \\ &\mathcal{C}'_i \leftarrow \pi_N(\mathcal{C}_i), \quad \mathcal{C}_i \leftarrow \mathcal{C}'_i, \quad \mathcal{C}_0 = \emptyset, \end{aligned} \tag{6}$$

where $k_{Bi:B(i+1)-1}$ and $v_{Bi:B(i+1)-1}$ are the keys and values selected from X_i respectively, and $\mathcal{C}_i = (K_i, V_i)$ is the KV cache after processing the first i prompt blocks. As in Equation (4), we concatenate the B new KVs to the current cache, apply π_N and update the cache in Equation (6).

We refer to this as *block prompt processing*. Its main advantage is the ability to control of the compute and memory overhead of KV cache management by adjusting the block size B and cache budget N. Note that, in a decoder-based architecture, applying block prompt processing to X with $B = T$ yields the same result as processing all of X at once and choosing $B = 1$ corresponds to the token generation phase of LLM evaluation.

Attention-Based Token Eviction Challenges Despite its advantages, block prompt processing introduces a challenge for KV cache eviction: eviction decisions in block X_i impact the cache used by X_{i+1}, causing eviction errors to compound over time. When the model processes X_i, attention-based eviction methods retain KVs with high attention weights derived from $X_0, \ldots, X_i$ rather than all of X, which may prematurely evict KVs with high weights in upcoming blocks.

3 Method

We demonstrate a negative correlation between attention scores and the cosine similarity among keys (Section 3.1) and leverage this observation to develop KeyDiff (Section 3.2), followed by a theoretical justification of KeyDiff (Section 3.3) and preliminary evidence of its efficacy (Appendix C.2).

3.1 Correlation of Attention Scores and Key Dissimilarity

To address the shortcomings of attention-based KV cache eviction in Section 2.4, we develop an alternative attention-free scoring metric that retains significant KVs across blocks while being resource efficient. We recall the "attention sink" phenomenon: LLMs often assign high attention weight to the first few tokens, regardless of the input [35, 29]; these highly weighted tokens are called *sink tokens*. However, the index of the sink tokens can vary across heads and layers and reside deeper in the sequence than the first few tokens. This observation motivates the following hypothesis: *high attention scores can be determined by the intrinsic properties of the keys rather than by any particular combination of keys and queries.*

Correlation of Key Similarity and Attention Scores We evaluate our hypothesis by inspecting the cosine similarities between keys computed inside an attention block. We visualize the pairwise cosine similarities between keys along with the attention weights in two particular heads and layers in Figure 2. We observe that keys with lower cosine similarity with other keys exhibit higher relative attention scores regardless of the choice of query, such as the 4th and 15th keys in Figure 2a, or the 1st key in Figure 2b. Pairwise cosine similarity of keys is solely a function of the keys in the cache, which are independent of input queries; the surprising aspect of Figure 2 is the negative correlation with attention weights. These distinctive keys essentially recover the attention sink phenomenon [35].

3.2 KeyDiff

Based on the observation in Section 3.1, we propose KeyDiff, which evicts tokens from the KV cache based on key similarity. If the cache $\mathcal{C}$ has intermediate size n and budget N where $n > N$, π_N^{KeyDiff} is defined as:

$$\begin{aligned} S &= \texttt{topk}(-\operatorname{CosSim}(K)\mathbf{1}, N), \\ K' &= \texttt{gather}(K, S), \quad V' = \texttt{gather}(V, S) \end{aligned} \tag{7}$$

where $K \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{n\times d}$ are the cached keys and values, $\operatorname{CosSim}(K) \in \mathbb{R}^{n\times n}$ is the pairwise cosine similarity matrix of keys in K with $\operatorname{CosSim}(K)_{ij} = \frac{k_i \cdot k_j}{\|k_i\|\|k_j\|}$, and $\mathbf{1} \in \mathbb{R}^n$ is a vector of ones.

Efficient Variant of KeyDiff Unlike attention-based eviction policies, KeyDiff does not require access to the attention weights A, facilitating optimized attention kernels that do not materialize A such as FlashAttention [8]. However, computing the pairwise cosine similarities runs in $\mathcal{O}(n^2)$ time. Fortunately, we can compute the score of each token in Equation (7) in $\mathcal{O}(n)$ as follows:

$$S = \texttt{topk}(-\operatorname{CosSim}(\mu(\hat{K}), \hat{k}_i), N) \tag{8}$$

where $\mu(\hat{K}) = \frac{1}{n}\sum_{i=1}^{n} \hat{k}_i$ and $\hat{k}_i = \frac{k_i}{||k_i||}$. We refer to $\mu(\hat{K})$ as the *anchor vector*. We show this formulation retains the same KVs of Equation (7) under a mild condition. (see Appendix C.2). Our experimentation has shown that the anchor vector $\mu(\hat{K})$ can be replaced with $\mu(K)$ without losing accuracy (see Table 15). We evaluate the efficient KeyDiff described in Figure 3 using unnormalized keys k in all subsequent sections. Figures 5 and 8 to 10 visualize the keys retained and evicted by sink attention [35], TOVA [24] and KeyDiff via PCA. KeyDiff retains more varied keys. A full complexity and FLOP analysis can be found in Appendices B and B.1

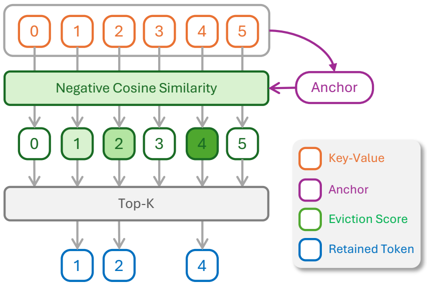


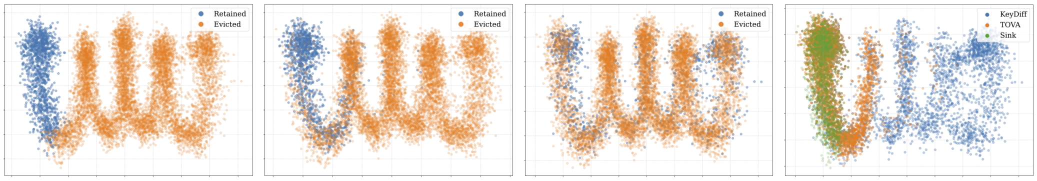
Figure 3: **An overview of KeyDiff. (1)** KeyDiff first computes the anchor vector by taking the average of the keys in the KV cache, **(2)** computes the cosine similarity between the keys and the anchor resulting in eviction scores whose color intensities indicate the score values, and **(3)** retains the KV pairs with the lowest similarities.

KeyDiff with Sliding Window In tasks such as reasoning and coding, where the most recent tokens are often important, we can augment KeyDiff and its efficient variant to use a percentage of the cache budget for a *sliding window* [6], which we call KeyDiff *with sliding window*. This extension introduces no complexity or memory overhead and we observe better results on certain tasks than vanilla KeyDiff (Table 14 and Appendix E).

3.3 Why KeyDiff Works: A Theoretical Perspective

To solidify the theoretical foundation of KeyDiff and show that KeyDiff ultimately selects keys most aligned with queries, we prove the following two results. We first validate the relationship between cosine similarity and attention scores observed in Figure 2 by bounding the attention score of a new incoming key $k^\star$ in terms cosine similarity with a fixed query q:

Lemma 3.1. *Suppose that for a fixed query token q, there is a set of key tokens $\{k_i\}_{i=1}^n$ such that $||k_i||_2^2 < M$, $\forall$ i. Without loss of generality suppose $||q|| = 1$ and assume k^* is a key not in $\{k_i\}_{i=1}^n$*



(a) Sink Attention (b) TOVA (c) KEYDIFF (d) Retained keys only

Figure 5: **(a, b, and c)** PCA Visualizations in two dimensions of a key cache managed with Sink, TOVA, and KEYDIFF. Retained tokens are blue, while evicted tokens are orange. Keys are taken from layer 5 and head 3 of Llama3.2-3B-Instruct, and generated using the NarrativeQA dataset. **(d)** PCA visualization of the retained keys for each KV cache eviction method.

with $||k^*||_2^2 < M$ *that has attention weight* $w > 0$*. Then, for* $n \to \infty$*,*

$$\frac{-\log(1-w)}{2M} - 1 \leq \mathit{CosSim}(k^*, q)$$

We then establish a relationship between the cosine similarities of $k^\star$, q, and the mean of prior keys $\bar{k}$:

Theorem 3.2. *Consider tokens* k^**,* q *as above, and the average of the keys tokens* $\bar{k}$*. Suppose* $\mathit{CosSim}(k^*, q) = \beta_q > 0$ *and* $\mathit{CosSim}(\bar{k}, q) = \alpha_q < 0$*. Then*

$$\mathit{CosSim}(\bar{k}, k^*) \leq 1 + \alpha_q\beta_q - 0.5\alpha_q^2 - 0.5\beta_q^2. \tag{9}$$

By combining Lemma 3.1 and Theorem 3.2, we establish a relationship between the attention weight w and the KEYDIFF score $\mathrm{CosSim}(\bar{k}, k^*)$. As $\mathrm{CosSim}(\bar{k}, q)$ decreases and $\mathrm{CosSim}(k^*, q)$ increases (along with the attention weight w), then $\mathrm{CosSim}(\bar{k}, k^*)$ tends to -1: this means KEYDIFF selects distinct keys most aligned with q. We visualize this in Figure 4 with a PCA embedding of keys and queries from a single head of Llama 3.2 3B, highlighting the relationship between top scoring keys via KEYDIFF, the anchor vector and queries. Similar trends are found from the other layers and heads as shown in Figure 11. The proofs of Lemma 3.1 and Theorem 3.2 are in Appendix C.3, along with empirical motivation for the chosen assumptions.

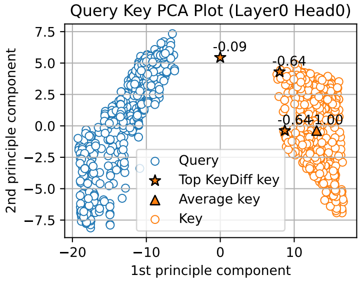
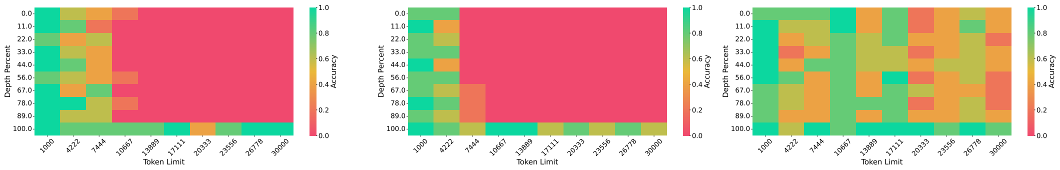


Figure 4: PCA embedding of keys and queries from Llama 3.2 3B

4 Experiments

In this section, we empirically demonstrate the effectiveness of KEYDIFF. We begin with a description of competing, state-of-the-art eviction methods, followed by a detailed description of the evaluation setup, then present our experimental results. Our findings can be summarized as follows:

- **Needle-In-a-Haystack.** KEYDIFF outperforms competing eviction policies on the Needle-In-A-Haystack benchmark (Section 4.1).
- **LongBench.** KEYDIFF outperforms competing eviction policies with block size $B = 128$ on LongBench, achieving an 1.5% accuracy drop with a 6K cache budget (∼33% compression rate) and ≤ .04% with a 8k cache budget (∼23% compression rate) with Llama-3.1-8B-Instruct and Llama-3.2-3B-Instruct (Section 4.2).
- **Reasoning.** KEYDIFF performs competitively on the Math-500 reasoning benchmark with other eviction methods using the DeepSeek-R1-Distill-Qwen-7B and Llama-8B, and shows near eviction-free baseline performance when augmented with a sliding window (Section 4.3) for DeepSeek-R1-Distill-Llama-8B.
- **Ablation Study.** We perform an ablation study on the main parameters of KEYDIFF and show that utilizing negative cosine similarity as the eviction criteria and the mean of cached keys as the anchor vector performs best. (Section 4.4).



(a) TOVA (b) SnapKV (c) KEYDIFF

Figure 6: Accuracy across document length and needle depth for needle in a haystack test. Cache size is 6K with $B = 128$.

- **Efficiency.** We compare the end-to-end inference latency of KEYDIFF, [20] and [24] and observe a 30% latency improvement with KEYDIFF (Section 4.5).

Experimental Setup We apply several cache eviction methods to several decoder-only transformer-based language models, including Llama 3.1-8B-Instruct [10], Llama 3.2-3B-Instruct [10], and Qwen 2.5-3B/7B-Instruct [38]. We evaluate these models using H2O [43], TOVA [24], SnapKV [20], and StreamingLLM [35], (or “sink attention") cache eviction policies, along with the eviction-free model as a baseline. We simulate a resource constrained environment by processing prompts and generating responses using Equation (6), with a block size of $B = 128$ for prompt processing and $B = 1$ for token generation using greedy decoding for all experiments. We denote the cache budgets of 2048, 4096, 6144 and 8192 as 2K, 4K, 6K and 8K, respectively.

4.1 Needle In a Haystack

To compare the impact of various cache eviction policies on fact retrieval, we conduct the “Needle In a Haystack" test [21, 16]. This test embeds specific information (“needle") at different points within a body of unrelated text (“haystack"); finding and retaining the needle is challenging for eviction policies, which can't know what information must be retained during block prompt processing. The results are shown in Figure 6 and Figure 19, where we show the recall accuracy of Llama3.2-3B-Instruct across different document lengths (x-axis) and needle depths (y-axis) with a cache size of 6K. KEYDIFF performs similarly to TOVA, SnapKV and sink attention for shorter documents and outperforms all three methods as the document length increases.

4.2 LongBench

LongBench [4] is a bilingual, multi-task benchmark suite for LLMs, providing a comprehensive stress test for long prompt inputs. LongBench is useful for evaluating cache eviction methods in a resource constrained environments with a fixed memory budget: 51% of prompts are longer than the largest KV cache size of 8K. For cache budgets of 6k and 8k tokens, prompts in LongBench are compressed by 33% and 23% respectively on average, (see Appendix F.3 for more detail.)

Table 1 summarizes the evaluation results of Llama 3.1-8B-Instruct and Llama 3.2-3B-Instruct on the English subset of LongBench with 2K, 4K, 6K, and 8K cache budgets using various eviction policies with block prompt processing enabled with $B = 128$. As shown in Table 1, KEYDIFF outperforms other eviction strategies across most tasks, even demonstrating better performance with smaller cache budgets. KEYDIFF shows significant a improvement on the PassageRetrieval-en (PR-en) dataset, which tests whether long-term dependencies within a long prompt can be correctly recognized [4], while achieving near full-context model performance even with the smallest budget. Adding a sliding window to KEYDIFF improves coding task performance (Table 14). We observed similar trends in the full LongBench task suite as shown in Table 11 and in the additional results in Appendix F.

KEYDIFF exhibits similar or better performance compared to competing methods. Notably, the attention-based methods (e.g., H2O, TOVA, and SnapKV) show significant performance improvements over the $B = 128$ case. This result supports our hypothesis: an eviction scheme robust to changes in the scope of comparison among tokens is essential in memory constrained environments where token-wise attention weight can't be fully materialized.

Additional Results We present the full evaluation results in Table 11 and more complete comparisons on LongBench in Appendix F, such as: standard prompt processing with a single large block

Table 1: **Llama-3.1-8B/3.2-3B-Instruct LongBench results with $B = 128$ (Higher is better)**. We highlight the best and second best methods within a given budget with **bold** and underline. We omit Chinese dataset results and other model results due to space limit. The full evaluation results are in Table 11. †: A subset of samples (183/200) were evaluated due to OOM errors.

		Single Doc. QA			Multi Doc. QA			Summarization			Fewshot Learning			Synthetic		Code		
		Narrative QA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	Avg.
Llama3.1-8B		30.05†	47.00	56.12	57.33	47.81	32.25	34.86	25.32	27.02	73.00	91.61	43.37	8.33	99.50	61.66	51.94	49.20
H2O	2K	1.74	21.15	25.33	26.11	24.15	8.78	2.17	2.70	16.78	44.00	29.36	7.62	2.25	5.88	40.15	12.14	16.89
	4K	4.07	36.16	36.00	33.52	32.87	17.78	6.66	5.95	24.09	55.00	47.65	17.41	4.00	24.50	54.85	21.43	26.37
	6K	8.52	43.31	44.80	40.03	42.46	21.68	11.85	8.78	26.03	62.00	56.39	25.72	5.75	45.50	58.62	29.53	33.19
	8K	13.85	44.94	47.81	43.64	44.90	23.65	18.78	11.35	26.49	69.50	69.05	33.41	5.25	62.50	59.74	36.26	38.20
TOVA	2K	22.57	37.26	39.43	45.74	34.48	14.77	28.87	21.17	26.95	62.50	90.73	42.74	0.00	18.00	62.68	52.48	37.52
	4K	22.68	44.55	47.87	46.76	44.54	20.56	30.95	22.13	26.96	61.50	90.56	43.27	3.00	43.50	61.62	53.40	41.49
	6K	24.59	45.93	53.92	55.09	47.43	25.07	32.33	24.10	27.00	68.50	90.81	43.89	4.25	67.00	61.50	52.39	45.24
	8K	24.86	46.78	54.83	54.52	49.00	26.40	33.44	24.76	27.00	71.00	91.11	43.29	6.25	87.00	61.49	51.79	47.09
Sink	2K	21.83	34.27	29.24	38.64	29.50	12.59	28.51	20.21	26.62	65.00	89.46	42.20	2.00	25.50	64.95	59.54	36.88
	4K	22.94	43.01	39.08	44.04	41.39	19.09	31.08	21.57	26.78	70.00	91.53	42.29	3.00	38.50	62.12	58.84	40.95
	6K	25.41	47.40	44.13	47.39	45.73	21.90	32.53	22.19	26.87	72.00	91.25	43.41	3.08	52.50	62.22	56.24	43.39
	8K	23.53	46.63	48.68	49.61	47.16	21.14	33.10	23.20	26.92	72.00	91.29	43.79	3.25	66.00	62.18	56.43	44.68
SnapKV	2K	21.81	37.22	37.19	46.10	35.42	16.53	29.83	21.05	26.77	61.00	88.84	42.56	4.03	51.50	62.37	51.45	39.60
	4K	24.79	44.22	47.30	48.49	46.73	20.55	32.19	22.68	26.95	67.50	90.98	43.14	5.17	89.50	61.44	51.20	45.18
	6K	24.10	45.57	50.44	53.12	48.41	24.27	33.43	23.53	27.03	71.50	92.28	43.58	5.25	98.00	61.32	52.16	47.12
	8K	25.15	46.55	53.39	56.00	48.75	27.82	33.67	24.85	27.01	72.50	91.78	43.54	5.08	100.00	61.48	51.41	48.06
KeyDiff	2K	26.64	41.73	50.99	51.59	46.47	22.84	29.02	23.86	26.76	66.50	85.92	39.26	3.17	96.00	59.17	39.42	**44.33**
	4K	28.70	45.62	56.06	54.58	49.31	28.25	32.30	25.03	27.07	70.00	90.85	42.84	4.21	99.00	60.80	48.00	**47.66**
	6K	29.90	46.33	55.11	56.80	49.50	31.52	33.44	24.58	26.98	72.00	90.99	43.10	5.27	99.50	61.40	49.70	**48.51**
	8K	33.57	46.77	55.48	56.87	49.37	30.88	34.17	25.12	27.01	72.50	92.28	42.81	5.83	99.50	61.48	50.90	**49.03**
Llama3.2-3B		23.76	40.23	50.09	50.69	42.29	26.84	33.09	24.30	25.21	72.50	90.11	42.58	3.00	96.50	56.22	56.52	45.87
H2O	2K	1.63	19.96	20.20	18.02	19.56	2.88	0.78	1.55	15.97	41.00	21.97	9.83	0.50	0.50	39.71	13.91	14.25
	4K	2.92	31.94	33.23	24.49	28.08	7.55	5.44	6.30	22.77	53.00	38.85	20.33	1.50	7.50	51.23	22.94	22.38
	6K	4.62	38.81	39.06	34.66	35.52	15.21	10.51	10.01	24.25	61.50	53.23	27.37	0.50	13.00	54.55	32.29	28.44
	8K	9.65	39.66	43.20	38.09	40.41	21.46	17.80	13.28	24.67	70.00	64.30	32.19	2.00	24.50	55.00	39.09	33.46
TOVA	2K	17.14	30.14	32.44	35.96	30.05	13.08	26.15	19.70	25.04	56.50	87.81	40.48	2.50	11.50	55.51	52.36	33.52
	4K	20.52	39.53	42.47	44.12	38.42	18.22	29.36	21.36	24.96	63.50	88.98	41.50	3.00	23.50	55.72	56.66	38.24
	6K	20.22	39.78	45.86	49.08	41.54	20.43	30.50	22.17	25.11	66.50	89.00	42.50	4.00	46.50	55.57	57.53	41.02
	8K	21.08	40.67	49.07	48.69	41.93	23.05	31.64	22.85	25.21	69.00	89.25	42.19	2.50	71.00	55.77	57.47	43.21
Sink	2K	16.85	30.69	26.58	33.26	25.27	13.82	26.74	19.15	25.15	65.00	86.17	40.79	1.50	19.50	56.65	52.73	33.74
	4K	19.46	38.61	36.22	41.97	35.84	13.37	29.34	20.19	25.06	71.00	88.06	41.31	2.50	35.50	56.48	52.43	37.96
	6K	19.33	40.29	37.95	46.48	40.29	15.31	30.43	21.35	25.14	71.50	88.93	42.04	3.50	47.00	56.55	54.11	40.01
	8K	20.15	40.02	41.94	48.15	42.24	16.01	31.64	22.10	25.20	73.00	89.26	42.37	3.50	62.50	56.86	56.63	41.97
SnapKV	2K	17.38	31.37	31.48	37.77	30.05	11.54	27.03	19.93	24.97	59.00	88.13	40.48	3.50	32.50	56.32	55.91	35.46
	4K	19.85	39.22	39.86	46.70	37.98	16.64	29.79	21.21	25.01	65.50	89.35	40.95	2.50	62.50	55.74	56.88	40.60
	6K	20.83	39.65	44.48	49.30	40.18	20.28	31.27	22.73	25.09	69.00	89.95	41.47	4.00	85.00	55.69	57.82	43.55
	8K	20.49	40.80	48.16	48.78	41.65	24.79	31.81	23.46	25.17	70.00	90.17	41.99	5.00	94.00	55.77	57.29	44.96
KeyDiff	2K	18.29	36.65	45.44	46.09	35.41	13.79	28.16	21.45	25.01	60.00	85.24	37.00	1.00	60.50	54.13	42.01	**38.14**
	4K	22.34	40.60	49.15	50.14	40.30	21.65	31.38	23.44	25.06	66.50	87.92	41.41	2.50	88.50	55.55	52.24	**43.67**
	6K	22.29	40.68	50.14	51.74	42.19	24.83	32.39	23.53	25.19	71.00	90.02	42.00	3.00	95.00	55.86	54.39	**45.27**
	8K	22.41	40.77	50.10	49.83	43.58	28.09	32.78	23.60	25.17	72.00	90.17	42.46	3.50	96.50	55.85	55.65	**45.78**

(i.e. $B = \infty$) in Table 10; eviction method performance with Qwen 2.5-3B/7B-Instruct in Table 12; performance behavior with block sizes $B = [64, 256]$ in Table 13; and performance on KeyDiff combined with a sliding window as described in Section 3.2. We also compare against the L_2-norm minimizing eviction method of [9] in Table 11.

4.3 Math-500 Reasoning Benchmark

Reasoning is an important long-context task for LLMs. Unlike other long-context use cases, reasoning typically involves a relatively short prompt followed by a long generation, which presents unique challenges for token eviction methods. To evaluate the effectiveness of token eviction methods, we apply KeyDiff and SnapKV to the DeepSeek-R1-Distill-Qwen-7B and Llama-8B distilled models [12], and assess their performance on the Math-500 reasoning benchmark [13]. Surprisingly, we found that Llama equipped with KeyDiff and a moderate KV cache budget performs comparably to, or slightly better than, the eviction-free baseline, while also outperforming SnapKV. We kindly refer the reader to Appendix E for additional details on the reasoning task evaluation.

4.4 Ablation Study

We evaluate the design choices of KeyDiff, including the similarity metrics and the choice of the anchor vector, and validate the efficacy of KeyDiff. We provide a full description of the test setup in Appendix G and summarize the findings here:

- KeyDiff anchor choice does not greatly impact benchmark accuracies (See Table 15).
- KeyDiff using cosine similarity as the distance metric outperforms other metrics (See Table 16)

4.5 Latency and Complexity

Additionally, in order to demonstrate that KeyDiff decreases end-to-end inference latency, we measured time to first token for the Llama 3.2 3B instruct model using different block prompt

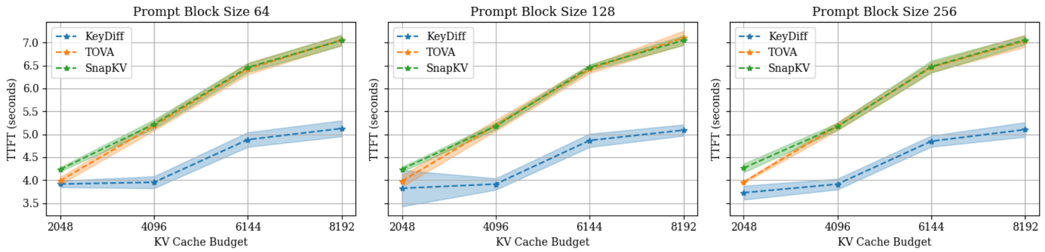


Figure 7: Time-to-first-token (TTFT) for Llama 3.2-3B using Flash Attention with different eviction strategies with block prompt processing sizes 64, 128, and 256.

processing sizes and cache strategies. These results are visualized in Figures 7 and 17. Since KEYDIFF does not require attention weight materialization, FlashAttention [8] can be used, resulting in up to 30% lower latency than TOVA and SnapKV. We compare the complexity of KEYDIFF with competitors in Appendix B and perform a complete FLOP count in Appendix B.1.

5 Related Work

Sparse Attention LLMs often exhibit high attention sparsity, where a small subset of keys receives a significant proportion of attention scores. This characteristic allows sparse approximation techniques to reduce the computational cost of attention. Similar to PagedAttention [18], Tang et al. [30] estimates the importance of a page (a contiguous set of keys) to a given query, whereas Rehg [26] further refined the budgets in a per-head manner. On the contrary, sample-based methods [44, 27] attempt to approximate token importance by inspecting the attention scores from the last few queries or certain query channel dimensions. Despite their effectiveness in reducing computational costs, these methods do not address the memory overhead of the KV cache, which typically retains all KVs.

KV Cache Compression Different approaches to compress the KV cache include architecture modification such as GQA [2], which shares a KV cache across a small number of heads. Other techniques to compress the KV cache include quantization such as in [15, 23, 42] in which the authors use various techniques to take advantage of existing patterns to efficiently quantize and compress the KV cache. More related to our work [40] uses a scoring mechanism to determine the precision of the quantization for different tokens.

Token Eviction Methods Unlike the sparse attention and KV cache compression methods, eviction methods *evict* KVs from the cache to reduce the size of the KV cache. As discussed in Section 2.3, the majority of the token eviction methods employ their own rules to decide the importance of the tokens by manipulating the attention score A. For example, by appropriately choosing the aggregation functions $\phi(A)$ of Equation (4), we can obtain existing attention-based eviction methods as discussed in Appendix A.1. Attention-based eviction may be a better choice when the entire prompt is being processed at once, as the eviction can be done by assessing the importance of all tokens simultaneously. However, computing the full attention score of long prompts could be prohibitively expensive in resource-constrained environments.

6 Conclusion

Inspired by our observation that distinctive keys tend to have high attention scores, we propose KEYDIFF, a training-free KV cache eviction method based on key similarity that enables large language models to operate in memory and compute constrained environments. We justify KEYDIFF by showing that it minimizes the pairwise cosine similarity among keys in the KV cache, maximizing the aforementioned diversity. KEYDIFF significantly outperforms state-of-the-art KV cache eviction methods under similar memory constraints, with only a 1.5% and 0.04% accuracy drop from the non-evicting baseline while achieving 33% and 23% KV cache memory reduction on LongBench. Similar to other token eviction methods, KEYDIFF is primarily designed and evaluated for the GQA attention mechanism used in models such as Llama and Qwen. In future work, we plan to extend KEYDIFF for seamless integration with other attention variants, such as Multi-Head Latent Attention [12].

References

[1] Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*, 2023.

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

[3] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*, 2023.

[4] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

[5] Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Razvan Pascanu, et al. Why do llms attend to the first token? *arXiv preprint arXiv:2504.02732*, 2025.

[6] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[9] Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. *arXiv preprint arXiv:2406.11430*, 2024.

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

[11] Nathan Godey, Éric de la Clergerie, and Benoît Sagot. Anisotropy is inherent to self-attention in transformers. *arXiv preprint arXiv:2401.12143*, 2024.

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

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

[14] Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, et al. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *arXiv preprint arXiv:2401.08671*, 2024.

[15] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[16] G. Kamradt. Needle in a haystack - pressure testing llms. GitHub repository, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

[19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[20] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[21] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[22] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

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

[24] Matanel Oren, Michael Hassid, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*, 2024.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[26] Isaac Rehg. Kv-compress: Paged kv-cache compression with variable compression rates per attention head. *arXiv preprint arXiv:2410.00161*, 2024.

[27] Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference. In *Forty-first International Conference on Machine Learning*.

[28] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[29] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.

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

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

[32] Mart van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*, 2024.

[33] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[35] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

[36] Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. Empowering 1000 tokens/second on-device llm prefilling with mllm-npu. *arXiv preprint arXiv:2407.05858*, 2024.

[37] Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. *arXiv preprint arXiv:2407.21018*, 2024.

[38] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

[39] Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. Pyramid-infer: Pyramid kv cache compression for high-throughput llm inference. *arXiv preprint arXiv:2405.12532*, 2024.

[40] June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

[41] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[42] Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *arXiv preprint arXiv:2405.03917*, 2024.

[43] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[44] Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, et al. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *CoRR*, 2024.

KeyDiff
Supplementary Material

A Extended Related Work

A.1 Attention-based eviction methods

In this section, we provide a unified framework to understand prominent attention-based eviction methods. As mentioned in Equation (5), we can specify attention-based eviction methods under the unified framework with proper selection of the aggregation function $\phi(A)$ as follows:

- TOVA [24]: $\phi^{\text{TOVA}}(A) = A_{-1,:}$,
- H2O [43]: $\phi^{\text{H2O}}(A) = A_{\text{prev}} + A^\top 1$,
- SnapKV [35]: $\phi^{\text{SnapKV}}(A) = (A^\top 1) * K$, where K is a vector of $\frac{1}{k}$ and k is the kernel size of average smoothing.

B Runtime and Memory Complexity

We analyze the runtime and memory complexity for the prominent KV cache eviction algorithms in Table 2. For a given block size B and cache budget N, KeyDiff requires $\mathcal{O}(N + B)$ runtime and memory. The same holds true for TOVA, since it only requires computing the bottom row of A. Sink attention retains the k first tokens in the input sequence, followed by a sliding window of size L, resulting in $\mathcal{O}(k + L) = O(N)$ memory and runtime, since $k + L$ equals the chosen cache budget. SnapKV computes attention over a sliding window of size L against $N + B$ keys from the incoming block and the key cache, so the memory and runtime complexity is $\mathcal{O}((N + B)L)$. H2O accumulates attention weights over all tokens, and computes the attention over the current block, so it will require $\mathcal{O}(NB + B^2)$ memory overhead and runtime. We summarize these details in Table 2

Table 2: **Runtime and memory complexity of token eviction methods.**

	Runtime Complexity	Memory Complexity
KeyDiff	$\mathcal{O}(N + B)$	$\mathcal{O}(N + B)$
TOVA	$\mathcal{O}(N + B)$	$\mathcal{O}(N + B)$
H2O	$\mathcal{O}(NB + B^2)$	$\mathcal{O}(NB + B^2)$
SnapKV	$\mathcal{O}((N + B)L)$	$\mathcal{O}((N + B)L)$
Sink	$\mathcal{O}(N)$	$\mathcal{O}(1)$

B.1 FLOP count of KeyDiff

The bulk of the computation in KeyDiff (neglecting the 'topk' operator) is the following two expressions:

- $\mu(\hat{K}) = \frac{1}{n} \sum_{i=1}^{n} \frac{k_i}{\|k_i\|}$
- $s_i = \text{CosSim}(\mu(\hat{K}), k_i) = \frac{\mu(\hat{K}) \cdot k_i}{\max(\|\mu(\hat{K})\| \cdot \|k_i\|, \epsilon)}, \quad i = 1, \ldots, n$

We will count the total number of additions, multiplications, square roots and divisions required by KeyDiff separately, since division and square root implementation are hardware-dependent, then assign weights to each operation at the end for a final count. Norms are assumed to be 2-norms. We count the FLOPs required for each operation as follows:

- $\|k_i\| = \sqrt{k_i \cdot k_i}$: since $k_i \in \mathbb{R}^d$: d multiplications, $d-1$ additions, one square root. Repeating for each i, this contributes nd multiplications, $n(d-1)$ additions, and n square roots.
- $\frac{k_i}{c}$: naively, d divisions, but can be rewritten as one division and d multiplications. Repeating for each i, this contributes nd multiplications and n divisions.
- $\frac{1}{n}\sum_{i=1}^{n} c_i$, for $c \in \mathbb{R}^d$: one division, $(n-1)d$ additions.

Combining the above, we can compute the anchor vector using $2nd$ multiplications, $2nd - n - d$ additions, n square roots and $n+1$ divisions.

To compute the cosine similarity score, we have from above that $\mu(\hat{K}) \cdot k_i$ requires nd multiplications and $n(d-1)$ additions. Also from above, we have that computing $\|\mu(\hat{K})\|$ requires d multiplications, $d-1$ additions and one square root. We reuse the computation of $\|k_i\|$ from the previous step and compute $\|k_i\|\|\mu(\hat{K})\|$ in n multiplications and $\max(\|k_i\|\|\mu(\hat{K})\|, \epsilon)$ with more n additions (assuming boolean comparison equals addition in cost). We can then divide through to compute $\frac{\mu(\hat{K}) \cdot k_i}{\max(\|k_i\|\|\mu(\hat{K})\|, \epsilon)}$ with n divisions. Therefore, computing the cosine similarity between the anchor and each key requires $nd + d + n$ multiplications, $nd + d - 1$ additions, $n+1$ divisions and one square root.

Adding everything up, KeyDiff requires:

1. $3nd + d + n$ multiplications,
2. $3nd - n - 1$ additions,
3. $2n + 2$ divisions,
4. $n + 1$ square roots,

If, based on x86 instruction tables, we declare additions and square roots cost one FLOP (i.e. can be computed in one cycle), multiplications cost three FLOPs, division is roughly 47 FLOPs, we arrive at a final FLOP count of:

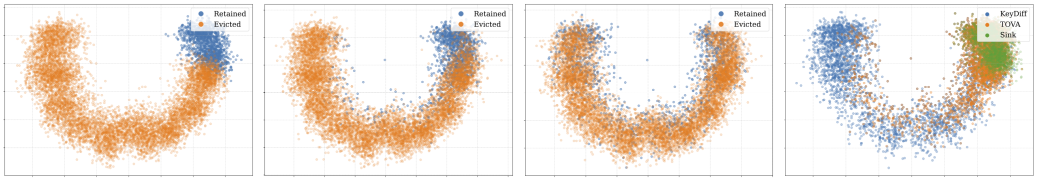
$$3(3nd + d + n) + (3nd - n - 1) + 47 * (2n + 2) + (n + 1) = (12d + 97)n + 3d + 94. \quad \text{(A.1)}$$

This is linear in the number of keys n with a small constant, relative to the quadratic complexity of the attention operator.

C KeyDiff: A Theoretical Perspective

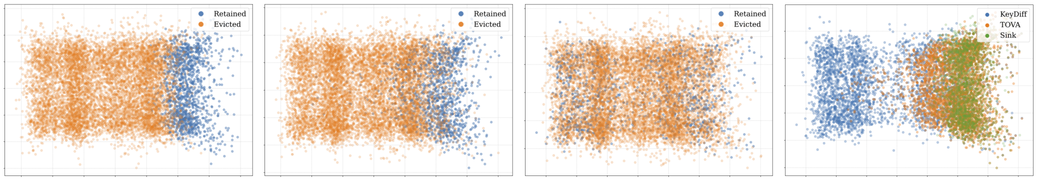
C.1 Additional PCA Visualizations

In order to demonstrate the phenomena in Figure 11 persists across all heads and layers, we have included several more visualizations as seen in Figure 8, Figure 9, and Figure 10.



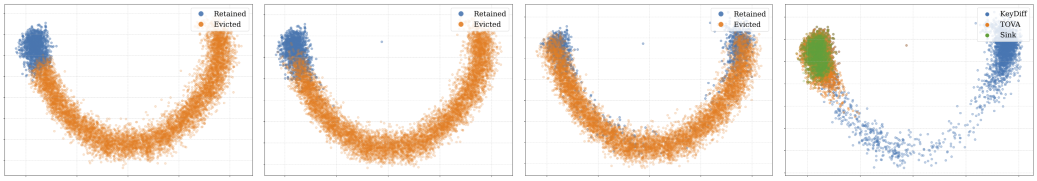
(a) Sink Attention (b) TOVA (c) KeyDiff (d) Retained keys only

Figure 8: **(a, b, and c)** PCA Visualizations in two dimensions of a key cache managed with Sink, TOVA, and KeyDiff. Retained tokens are blue, while evicted tokens are orange. Keys are taken from layer 27 and head 4 of Llama3.2-3B-Instruct, and generated using the NarrativeQA dataset. **(d)** PCA visualization of the retained keys for each KV cache eviction method



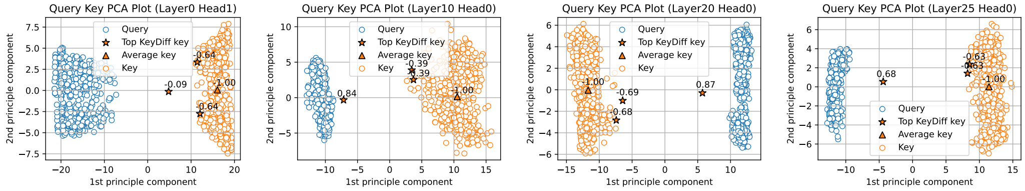
(a) Sink Attention (b) TOVA (c) KEYDIFF (d) Retained keys only

Figure 9: Keys taken from layer 20 and head 0 of Llama3.2-3B-Instruct



(a) Sink Attention (b) TOVA (c) KEYDIFF (d) Retained keys only

Figure 10: Keys taken from layer 8 and head 1 of Llama3.2-3B-Instruct



(a) Layer 0 Head 1 (b) Layer 10 Head 1 (c) Layer 20 Head 1 (d) Layer 25 Head 1

Figure 11: PCA plots of Query and Keys from Llama-3.2-3B-Instruct

C.2 Derivation of KEYDIFF from an Optimization Perspective

To leverage the observation in Section 3.1, we minimize the sum of pairwise cosine similarities of each key retained in the cache. This can be formulated as a constrained optimization problem with the keys $K \in \mathbb{R}^{n \times d}$ whose element is k_i, and budget N smaller than n:

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \sum_{i \in S} \sum_{j \in S} \frac{k_i \cdot k_j}{\|k_i\| \|k_j\|} \\ \text{subject to} \quad & S \subseteq \{1, \dots, n\}, \\ & |S| = N \end{aligned} \tag{A.2}$$

This is a combinatorial optimization problem, which is difficult to solve efficiently, particularly during inference. However, we can relax Equation (A.2) to produce a more tractable approximation to the original problem. First, we rewrite Equation (A.2) by normalizing keys k_i such that $\hat{k}_i = \frac{k_i}{\|k_i\|}$ resulting in:

$$\sum_{i \in S} \sum_{j \in S} \hat{k}_i \cdot \hat{k}_j = \sum_{i \in S} \hat{k}_i \cdot \left(\sum_{j \in S} \hat{k}_j \right) = \sum_{i \in S} \langle \hat{k}_i, N\mu(\hat{K}_S) \rangle$$

where $\mu(\hat{K}_S) = \frac{1}{N} \sum_{j \in S} \hat{k}_j$ is the empirical mean of normalized keys in S. This objective requires recomputing $\mu(\hat{K}_S)$ for each candidate subset S. The sub-sampled mean tends to converge to the mean over the entire set, the problem can be further relaxed by replacing $\mu(\hat{K}_S)$ with $\mu(\hat{K}) =$

$\frac{1}{n}\sum_{i=1}^{n}\hat{k}_i$. Dropping N from the objective (since it doesn't affect the solution) yields:

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \sum_{i \in S} \hat{k}_i \cdot \mu(\hat{K}) \\ \text{subject to} \quad & S \subseteq \{1, \dots, n\}, \\ & |S| = N \end{aligned} \tag{A.3}$$

The optimal solution of Equation (A.3) can be found by sorting tokens using their cosine similarity with $\mu(\hat{K})$ and selecting the smallest N, leading to the algorithm described in Equation (8).

Key Cache Diversity and KEYDIFF To empirically verify KEYDIFF's ability to retain diverse keys, we apply PCA to the keys computed in an attention block of Llama 3.2-3B-Instruct after evaluating a long context prompt. We visualize the distribution of retained and evicted keys by applying sink attention [35], TOVA [24], and KEYDIFF as eviction policies in Figure 11. Visual observation reveals the tokens retained by sink attention and TOVA tend to tightly cluster together while KEYDIFF's retained tokens are more evenly distributed. This observation generalizes across heads and layers, as shown in Appendix C.1.

We also visualize the log determinant of the Gram matrix of the key cache $\log(\det(KK^T))$ generated using different eviction policies in Figure 12. This quantity corresponds to the volume of space spanned by the keys in $\mathcal{C}$. The distribution of volumes for KEYDIFF attains higher values, indicating that the retained keys are more distinctive than TOVA and sink, corroborating the results of Figure 11. Details of how these plots were generated are discussed in Appendix F.1.

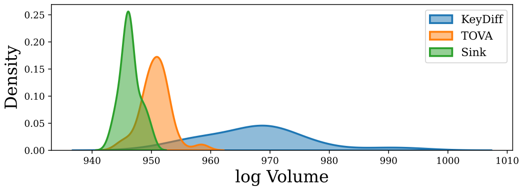
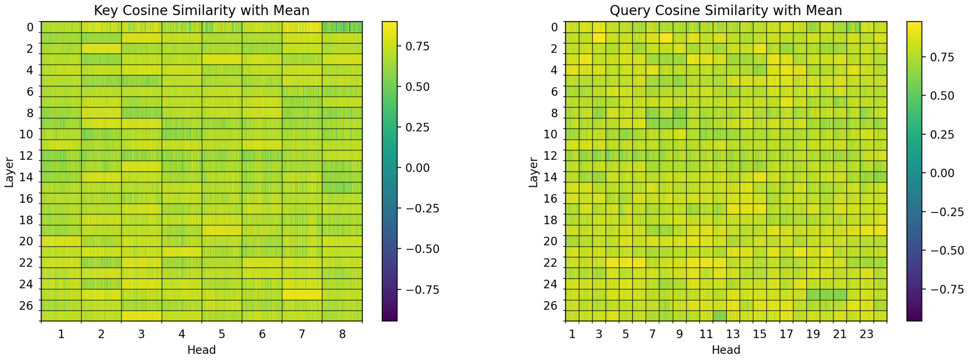


Figure 12: Distribution of $\log\left(\det\left(KK^T\right)\right)$ from the Qasper dataset in LongBench. Larger values mean more of the key space is spanned by the key cache. KEYDIFF retains keys that span a greater volume of the ambient space than TOVA or sink attention.

C.3 Attention Sinks and Approximate Collinearity



(a) Keys

(b) Queries

Figure 13: Cosine similarity between keys and their mean (left) and queries and their mean (right) across heads and layers

To better understand why there exists a negative correlation between cosine similarity between keys and attention scores, we look to recent research that seeks to the importance of attention sinks in decoder-based LLMs. The authors in [5] show that attention sinks emerge from training decoder-based LLMs since they can denoise the model and prevent rank collapse by limiting over mixing in attention heads. Moreover, attention patterns in decoder based models demonstrate that most

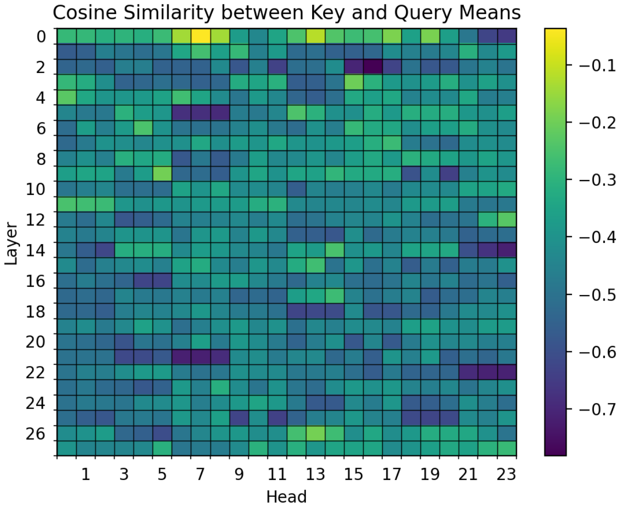


Figure 14: Cosine similarity of mean key and mean query for each head and layer.

attention logits are quite small (and almost always negative) for most keys and queries. This allows the attention sink to have high attention activation, preventing over mixing in addition to allowing the heads to specialize and selectively identify important tokens.

At the same time, the results in [11] suggest that hidden states, keys and queries are all approximately collinear in the sense that $\text{CosSim}(x_i, x_j) \gg 0$. In geometric terms, this means that most key tokens and query tokens lie in the same direction in Euclidean space. Our own results, seen in Figure 13 demonstrate that this is the case for both keys and queries. Our experiments show that most keys and queries lie within a small angular distance from the mean key and query. More than this, we see that across all heads, the mean key and mean query have negative cosine similarity Figure 14. Moreover, as seen in Figure 15, we find that the norms of keys and queries are tightly clustered around a relatively fixed value. This means that variations in the norm of key and query tokens have less impact on the magnitude of attention scores than their direction. These three observations indicate that most keys and queries combine to create uniformly small attention logits, and that larger attention weights are constructed by projecting keys closer to the direction of the mean query. This appears to be the fundamental mechanism through which over mixing is prevented: if most attention activations are very small, each head can increase the activations of a small number of keys across most queries selectively projecting them to be more aligned with the distribution of queries. This hypothesis is further supported by the fact that sink tokens themselves often have small norm, which results in an approximate no-op in the attention head as in [5], however in this case, the only way for a key corresponding to a sink token to have high attention scores is if it is as parallel as possible to the set of query tokens.

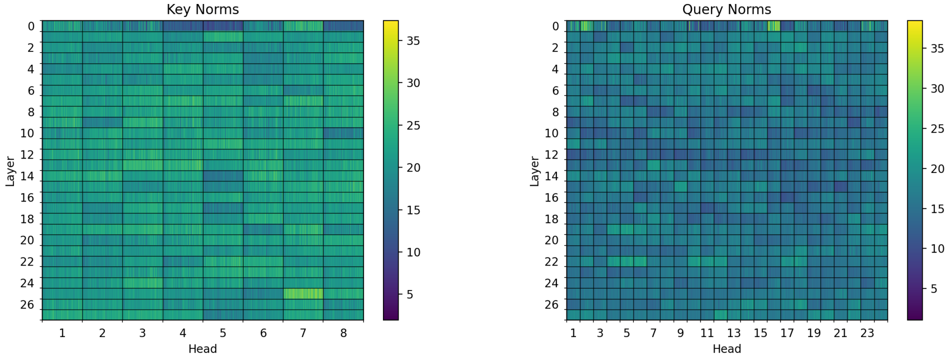


Figure 15: Distribution of L^2 norms for keys and queries across heads and layers.

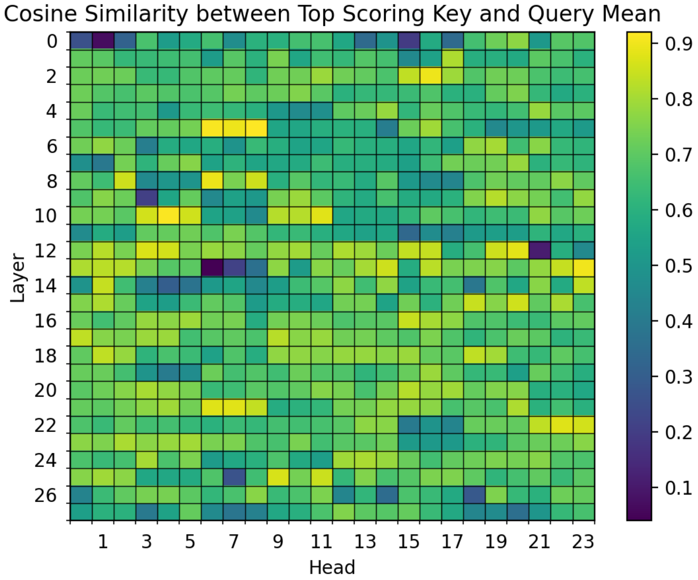


Figure 16: Cosine similarity between highest KEYDIFF scoring key token with mean query.

To verify the above hypothesis, we show Figure 16 that keys which have maximum angular difference from the mean key are aligned with the mean query, resulting in very large attention weights. This demonstrates how LLMs exploit the geometry of the hidden states and projections to limit over mixing, and selectively identify important tokens.

To summarize, we have for all attention heads in decoder based transformer models:

- *The majority of keys and queries are approximately collinear with their mean.*
- *Mean keys and mean queries have negative cosine similarity across all heads.*
- *Most keys and queries have a similar L2 norm.*
- *Decoder based attention heads can selectively increase attention weights for a fixed key by aligning it with the mean query.*
- *Key token importance can hence be measured by the angular distance between a key and the mean key.*

We can show mathematically with some reasonable assumptions based on the above observations that key tokens with persistently high attention scores must be geometrically aligned with queries.

Theorem C.1. *Suppose that for a fixed query token q, there is a set of key tokens $\{k_i\}_{i=1}^n$ such that $||k_i||_2^2 < M,\ \forall\ i$. Without loss of generality suppose $||q|| = 1$, the scaling parameter is 1 and assume k^* is a key not in $\{k_i\}_{i=1}^n$ with $||k^*||_2^2 < M$ that has attention weight $w > 0$:*

$$w = \frac{\exp({k^*}^\top q)}{\exp({k^*}^\top q) + \sum_{i=1}^{n} \exp(k_i^\top q)}.$$

Then

$$\frac{\log(\frac{n}{n+1}) - \log(1-w)}{2M} - 1 \leq \mathit{CosSim}(k^*, q)$$

Proof. To show this we have that

$$\begin{aligned}
w &= \frac{\exp({k^*}^\top q)}{\exp({k^*}^\top q) + \sum\limits_{i=1}^{n} \exp(k_i^\top q)} \\
w\left(\exp({k^*}^\top q) + \sum_{i=1}^{n} \exp(k_i^\top q)\right) &= \exp({k^*}^\top q) \\
w \sum_{i=1}^{n} \exp(k_i^\top q) &= (1-w)\exp({k^*}^\top q)
\end{aligned}$$

Note that $-M \leq k_i^\top q \leq M$ and hence

$$\begin{aligned}
w \sum_{i=1}^{n} \exp(-M) &\leq (1-w)\exp({k^*}^\top q) \\
wn\exp(-M) &\leq (1-w)\exp({k^*}^\top q) \\
\frac{wn\exp(-M)}{1-w} &\leq \exp({k^*}^\top q) \\
\log(wn) - M - \log(1-w) &\leq M\text{CosSim}(k^*, q) \\
\frac{\log(wn) - M - \log(1-w)}{M} &\leq \text{CosSim}(k^*, q) \\
\frac{\log(\frac{\exp({k^*}^\top q)}{\exp({k^*}^\top q) + \sum\limits_{i=1}^{n} \exp(k_i^\top q)} n) - M - \log(1-w)}{M} &\leq \text{CosSim}(k^*, q) \\
\frac{\log(\frac{\exp({k^*}^\top q)}{(n+1)\exp(M)} n) - M - \log(1-w)}{M} &\leq \text{CosSim}(k^*, q) \\
\frac{-M\text{CosSim}(k^*, q) + \log(\frac{n}{n+1}) - 2M - \log(1-w)}{M} &\leq \text{CosSim}(k^*, q) \\
\frac{\log(\frac{n}{n+1}) - 2M - \log(1-w)}{M} &\leq 2\text{CosSim}(k^*, q) \\
\frac{\log(\frac{n}{n+1}) - \log(1-w)}{2M} - 1 &\leq \text{CosSim}(k^*, q)
\end{aligned}$$

Taking the limit as $n \to \infty$ produces Lemma 3.1 □

The above proof demonstrates that as long as the norms of the keys are bounded, in order for an attention head to be able to freely allocate w attention weight to a given key k^*, the cosine similarity between k^* and q must be high, therefore k^* and q must be approximately collinear.

Using this result, we can also show that as long as the cosine similarity between k^* and q is high, while the cosine similarity between $\bar{k}$ and q is low, CosSim$(k^*, \bar{k})$ is small. Note that, since empirical results demonstrate that most keys have high cosine similarity with their mean $\bar{k}$, a key with high importance, approximately collinear to q, will also have low cosine similarity to $\bar{k}$. Generally, this also suggests that key tokens with low cosine similarity to $\bar{k}$ have greater importance.

In order to show this, we need the following auxiliary result.

Lemma C.2. *Suppose $\{x_1, ..., x_n\}$ is an orthonormal basis of $\mathbb{R}^n$ and $y \in \mathbb{R}^n$. Define $\alpha_i =$ CosSim(x_i, y). Then $\sum\limits_{i=1}^{n} \alpha_i^2 = 1$.*

Proof. Note that $\alpha_i = \frac{y^\top x_i}{||y||||x_i||} = \frac{y^\top x_i}{||y||}$. If we expand $\frac{y}{||y||}$ in the basis $\{x_1, ..., x_n\}$ we see that

$$\begin{aligned} \frac{y}{||y||} &= \sum_{i=1}^{n} \left(\frac{y}{||y||}^\top x_i \right) x_i \\ &= \sum_{i=1}^{n} \alpha_i x_i \end{aligned}$$

But then, we know that since $\left|\left| \frac{y}{||y||} \right|\right|_2^2 = 1$, then $\left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{i=1}^{n} \alpha_i x_i \right\rangle = 1$. But we have

$$\begin{aligned} \left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{i=1}^{n} \alpha_i x_i \right\rangle &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \alpha_i x_i, \alpha_j x_j \rangle \\ &= \sum_{i=1}^{n} \langle \alpha_i x_i, \alpha_i x_i \rangle \\ &= \sum_{i=1}^{n} \alpha_i^2 \end{aligned}$$

proving the result. □

Theorem C.3. *Consider tokens k^*, q, $\bar{k}$ as above where $\bar{k}$ is the average of the keys tokens. Suppose CosSim$(k^*, q) = \beta_q > 0$ and CosSim$(\bar{k}, q) = \alpha_q < 0$. Then CosSim$(\bar{k}, k^*) \leq 1 + \alpha_q \beta_q - \frac{1}{2}\alpha_q^2 - \frac{1}{2}\beta_q^2$.*

Proof. Consider the cosine similarity of $\bar{k}$ and k^*:

$$\text{CosSim}(\bar{k}, k^*) = \frac{{k^*}^\top \bar{k}}{||k^*||||\bar{k}||}$$

expand $\bar{k}$ in an orthonormal basis which contains q, $\{q, r_1, ..., r_{n-1}\}$ such that

$$\bar{k} = ||\bar{k}|| \left(\alpha_q q + \sum_{i=1}^{n-1} \alpha_i r_i \right)$$

where $\alpha_i = \text{CosSim}(\bar{k}, r_i)$. Additionally, define $\beta_i = \text{CosSim}(k^*, r_i)$ and note that by the definition of an orthonormal basis and the cosine similarity operation, using the result from Lemma C.2 we have that $\alpha_q^2 + \sum_{i=1}^{n-1} \alpha_i^2 = 1$ and that $\beta_q^2 + \sum_{i=1}^{n-1} \beta_i^2 = 1$. Now we have that

$$\begin{aligned} \frac{{k^*}^\top \bar{k}}{||k^*||||\bar{k}||} &= \frac{{k^*}^\top \left(||\bar{k}||\alpha_q q + \sum_{i=1}^{n-1} ||\bar{k}||\alpha_i r_i \right)}{||k^*||||\bar{k}||} \\ &= \alpha_q \beta_q + \frac{1}{||k^*||} \sum_{i=1}^{n-1} \alpha_i {k^*}^\top r_i \\ &= \alpha_q \beta_q + \sum_{i=1}^{n-1} \alpha_i \beta_i \\ &\leq \alpha_q \beta_q + \sum_{i=1}^{n-1} |\alpha_i||\beta_i| \end{aligned}$$

Table 3: Spearman correlation (ρ) between negative key cosine similarity and attention scores for each transformer layer of Llama-3.2-3B-Instruct.

Layer	1	2	3	4	5	6	7
ρ	0.8997	0.9561	0.9332	0.9496	0.9517	0.9484	0.9587
Layer	8	9	10	11	12	13	14
ρ	0.9578	0.9534	0.9570	0.9628	0.9554	0.9658	0.9578
Layer	15	16	17	18	19	20	21
ρ	0.9477	0.9538	0.9280	0.9373	0.9328	0.9340	0.9311
Layer	22	23	24		Mean ± Std		
ρ	0.9140	0.9060	0.8950		0.94 ± 0.02		

Applying Young's inequality we obtain

$$\begin{aligned}
&\leq \alpha_q\beta_q + \frac{1}{2}\sum_{i=1}^{n-1}\alpha_q^2 + \beta_q^2 \\
&= \alpha_q\beta_q + \frac{1}{2}(1-\alpha_q^2) + \frac{1}{2}(1-\beta_q^2) \\
&= 1 + \alpha_q\beta_q - \frac{1}{2}\alpha_q^2 - \frac{1}{2}\beta_q^2
\end{aligned}$$

□

Note that on the domain $\alpha_q \in [-1, 0)$, $\beta_q \in (0, 1]$ the function $1 + \alpha_q\beta_q - \frac{1}{2}\alpha_q^2 - \frac{1}{2}\beta_q^2$ is bounded above by 1 and decreasing to $-\frac{1}{2}$ as $\alpha \to -1$. Hence, the smaller $\text{CosSim}(\bar{k}, q) = \alpha_q$ is, the smaller $\text{CosSim}(\bar{k}, k^*)$ must be.

C.4 Correlation Analysis

We report the Spearman rank correlation between key cosine similarity and attention scores for each layer of the Llama-3.2-3B-Instruct model. Correlations are averaged over randomly sampled LongBench-Musique prompts. From Table 3, we observe a consistently high correlation ($\rho \approx 0.94$ on average) across all layers, indicating that geometrically distinctive keys (i.e., those with low pairwise cosine similarity) are strongly aligned with tokens receiving higher attention scores. This empirical evidence supports our theoretical claim in Section 3.3 that key diversity serves as a reliable proxy for token importance.

D TTFT Analysis

We have measured end-to-end inference latency (measured as time to first token (TTFT)) for Llama 3.2-3B via the standard Huggingface API when the model is using eager attention and FlashAttention as in Figure 17 and Figure 7. Tests are performed on NVIDIA A100 80GB GPUs. We test various block sizes and KV cache budgets. KeyDiff outperforms TOVA and SnapKV with FlashAttention as well as with eager attention for large cache budgets. We can see inference latency with KeyDiff is independent of block size when using FlashAttention because of KeyDiff's linear complexity and its lack of required materialized attention weights.

E Math-500 reasoning benchmark

In order to measure the effectiveness of different caching methods on reasoning tasks, we tested several different model using various caching algorithms on the Math 500 reasoning benchmark. Specifically, we test KeyDiff with a sliding window whose window size is 20% of the KV cache budget, and SnapKV on the DeepSeek-R1-Distill-Llama-8B and DeepSeek-R1-Distill-Qwen-7B

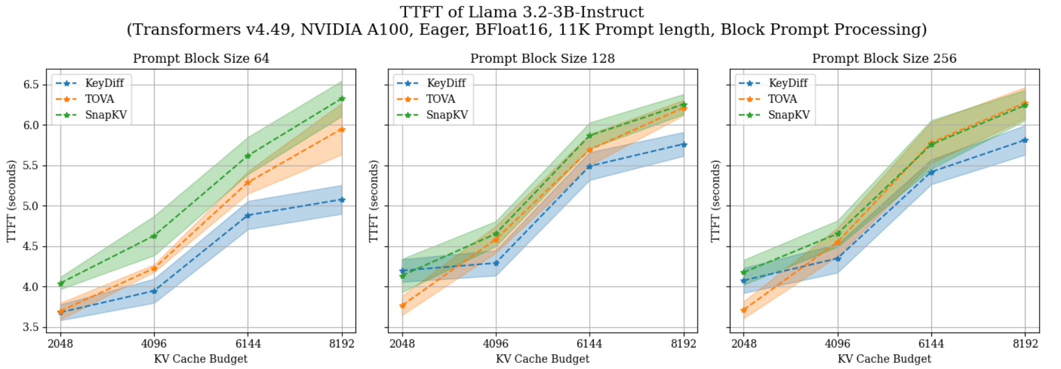


Figure 17: TTFT for eager attention with different cache eviction strategies using block size 64, 128, 256 for block prompt processing.

models. We randomly sample 5 responses with TopP $= 0.95$, Temperature $= 0.95$ with the 4096, 6144, and 32,786 max generation lengths. The reported scores are the average of accuracies over the random samples.

Table 4 summarizes the Math-500 evaluation results for Llama-8B. As shown in the table, token eviction methods generally perform well even with KV cache budgets that are strictly smaller than the maximum generation length. Surprisingly, we found that KeyDiff slightly outperforms baseline methods in certain configurations (e.g., KeyDiff with a 2K budget for 4K generation length, and KeyDiff with a 4K budget for 8K generation length).

To further analyze the effectiveness of token eviction, we measure accuracy in cases where the context length of the baseline method (i.e., prompt length + generation length) exceeds the available KV cache budget. As shown in Table 5, for samples where eviction is actively triggered, KeyDiff continues to outperform the token eviction baseline (SnapKV), and often achieves accuracy close to or slightly better than the non-evicting baseline.

We also conducted a similar evaluation on DeepSeek-R1-Distill-Qwen-7B and observed a slight performance degradation for token eviction methods compared to full KV cache baselines (See Tables 6 and 7.) However, KeyDiff still demonstrates comparable performance to SnapKV overall. This discrepancy may stem from architectural differences that Llama uses a lower GQA [2] ratio than Qwen, which results in more information compression in the KV cache. We hypothesize that models with more compression like Qwen are more sensitive to eviction since each evicted token contains more information in Qwen than Llama by design.

F Additional discussion for LongBench

F.1 Empirical Motivation for KeyDiff Setup

To generate Figure 11, we used the first sample from the test split of the narrativeqa task in LongBench to prefill the KV cache of Llama3.2-3B-Instruct with a block size of $B = 128$. The KV cache had a maximum size of 4096 while the sample was much longer, requiring KV eviction. We applied PCA to the key cache and repeated the process for sink attention, TOVA and KeyDiff.

To construct Figure 12, we sample 100 prompts from the `Qasper` dataset in LongBench [4], compute the log determinant of KK^T of the keys in the KV caches of each head and layer of Llama 3.2-3B-Instruct using a cache budget of $N = 2048$ and a block size of $B = 128$, and plot the distribution in Figure 12. We show this key distribution for sink attention, TOVA and KeyDiff.

F.2 Experiment Setup

In this subsection, we provide the experimental setup for KeyDiff and the baselines for the LongBench experiments. The LongBench evaluation is conducted using the default parameters of the LongBench evaluator with predefined prompt templates. Tests are performed on NVIDIA A100 80GB GPUs.

Table 4: **Math 500 results on DeepSeek-R1-Llama-8B distilled model (Higher is better)**. We highlight the methods showing the best performance within a given budget with **boldface**.

Method	Max Gen. Length	Budget	Flex Match	Exact Match	Avg. Gen. Length
Full	4K	N/A	0.711	0.537	2769
KeyDiff	4K	1024 2048	**0.695** **0.720**	0.531 0.546	2753 2740
SnapKV	4K	1024 2048	0.689 0.714	0.529 0.544	2759 2757
Full	8K	N/A	0.840	0.628	3812
KeyDiff	8K	2048 4096	**0.819** **0.844**	0.617 0.634	3888 3805
SnapKV	8K	2048 4096	0.805 0.828	0.610 0.627	3898 3898
Full	32K	N/A	0.898	0.668	6869
KeyDiff	32K	2048 4096 8192	**0.883** **0.894** **0.894**	0.662 0.668 0.667	7678 7312 7096
SnapKV	32K	2048 4096 8192	0.849 0.884 0.893	0.641 0.661 0.665	7509 7218 7005

For TOVA, H2O, and SnapKV, the set of attention weights computed from a single key cache due to grouped query attention [2] is aggregated by taking the average over the attention weights. Additionally, only for SnapKV, we apply average smoothing to the attention score with a kernel size of 7 and keep the most recent 32 tokens in the cache, following the suggestion of the original paper. For Sink, we used the first four tokens as the attention sink, following the suggestion of the original paper.

F.3 Longbench dataset statistics

In this section, we provide the length statistics of the Longbench Benchmark and in-depth analysis of compression ratios for the given KV cache budgets, such as 2K, 4K, 6K, and 8K.

Prompt lengths We measure the number of tokens in the samples using LLama tokenizer [31]. As shown in Figure 18, LongBench exhibits variability in sample length from the datasets.

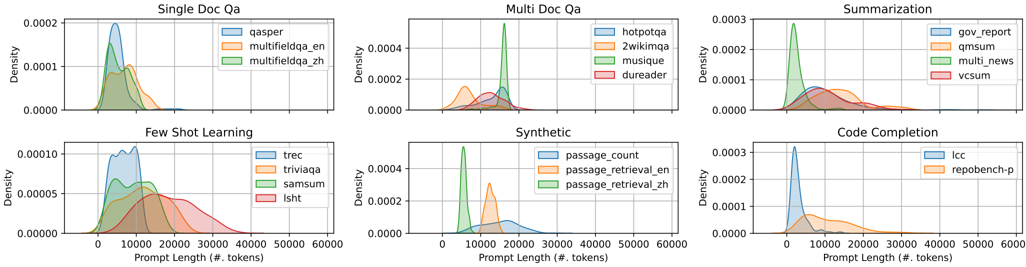


Figure 18: **Histograms of sample lengths measured by number of tokens**

Compression ratio The majority of other KV cache eviction studies assume an unconstrained memory footprint. Before they compress the cache by applying an eviction policy, they first set the target compression ratio and evict the appropriate number of KV pairs to satisfy the compression ratio [43, 24, 20]. On the other hand, we fix the KV cache size and ensure the number of cached tokens is less than or equal to the predefined cache size. Due to these differences, it is less straightforward to

Table 5: **Math 500 results on DeepSeek-R1-Llama-8B distilled model (Higher is better)**. We highlight the methods showing the best performance within a given budget with **boldface**.

Max Gen. Length = 4K				Max Gen. Length = 8K			
Num Tokens > 1K (497/500 samples)				Num Tokens > 2K (353/500 samples)			
	Budget	Flex	Exact		Budget	Flex	Exact
Full	N/A	0.709	0.534	Full	N/A	0.783	0.584
KeyDiff	1024	**0.693**	**0.528**	KeyDiff	2048	**0.756**	**0.570**
SnapKV	1024	0.687	0.526	SnapKV	2048	0.736	0.560
Num Tokens > 2K (240/500 samples)				Num Tokens > 4K (195/500 samples)			
Full	N/A	0.604	0.449	Full	N/A	0.650	0.455
KeyDiff	2048	**0.618**	**0.463**	KeyDiff	4096	**0.662**	**0.470**
SnapKV	2048	0.610	0.458	SnapKV	4096	0.637	0.453

Max Gen. Length = 32K			
Num Tokens > 2K (353/500 samples)			
	Budget	Flex	Exact
Full	N/A	0.783	0.584
KeyDiff	2048	**0.756**	**0.570**
SnapKV	2048	0.736	0.560
Num Tokens > 4K (195/500 samples)			
Full	N/A	0.650	0.455
KeyDiff	4096	**0.662**	**0.470**
SnapKV	4096	0.637	0.453
Num Tokens > 8K (162/500 samples)			
Full	N/A	0.793	0.550
KeyDiff	8192	**0.786**	**0.545**
SnapKV	8192	0.782	0.541

set appropriate KV cache budgets to satisfy the target compression ratios. Instead, we provide the average compression ratio, which is defined as:

$$\text{Average Compression Ratio} = \frac{1}{I}\sum_{i=1}^{I}\frac{N}{L_i},$$

where N is the KV cache budget, and L_i is the length of the i-th prompt (sample). We replace the summand with 1 whenever $N \geq L_i$, as compression doesn't occur in that setting.

As summarized in Table 9, 2K cache budgets have a 0.31 average compression ratio, which indicates 69% of input prompts are compressed. Our largest cache budget, 8K, exhibits a 0.77 average compression ratio.

F.4 Additional Results

G Ablation study

Selecting the Anchor Vector We have mainly evaluated KeyDiff using the method described in Equation (7). Scores to determine eviction are measured via cosine similarity with an *anchor vector* which can be computed in several ways. We run LongBench on Llama3.2-3B-Instruct with eviction

Table 6: **Math 500 results on DeepSeek-R1-Qwen-7B distilled model (Higher is better)**. We highlight the methods showing the best performance within a given budget with **boldface**.

Method	Max Gen. Length	Budget	Flex Match	Exact Match	Avg. Gen. Length
Full	4K	N/A	0.764	0.579	2630
KeyDiff	4K	1024	0.666	0.512	2692
		2048	**0.749**	0.570	2629
SnapKV	4K	1024	**0.692**	0.533	2655
		2048	**0.749**	0.566	2637
Full		N/A	0.877	0.658	3287
KeyDiff	8K	2048	0.811	0.613	3348
		4096	0.867	0.647	3208
SnapKV	8K	2048	**0.812**	0.612	3328
		4096	**0.868**	0.647	3214
Full	32K	N/A	0.923	0.682	4051
KeyDiff	32K	2048	0.811	0.613	4322
		4096	**0.897**	0.647	3800
		8192	**0.891**	0.663	3741
SnapKV	32K	2048	**0.812**	0.612	4279
		4096	0.868	0.647	3828
		8192	**0.891**	0.662	3808

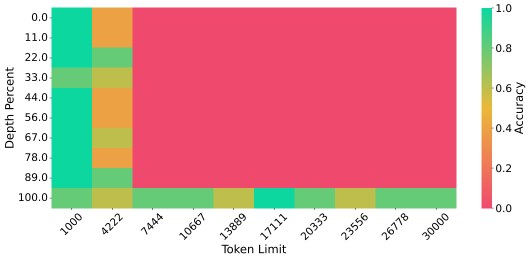


Figure 19: Needle in a Haystack results for Sink Attention [35]

policies using the following anchor choices: pairwise cosine similarity from Equation (7), denoted Pairwise; KeyDiff, using the mean of all normalized keys as an anchor, and using the median of keys as an anchor, denoted Median. Table 15 summarizes the average LongBench accuracy for the different methods to Llama 3.2-3B-Instruct. KeyDiff shows similar average scores to Pairwise. Additionally, KeyDiff and Median show similar scores, demonstrating that KeyDiff is robust to the selection of the anchor design.

Selecting the Similarity Metric We use cosine similarity as the scoring metric for eviction in KeyDiff based on our discussion in Section 2.2. This could be replaced with other metrics like the dot product or Euclidean distance. We evaluate KeyDiff variants using dot product and Euclidean distance as the similarity metric, denoted DotProd and Euclidean respectively, and report the results in Table 16. KeyDiff and DotProd show similar performance for 6K and 8K budgets. However, KeyDiff outperforms DotProd for smaller cache sizes. This implies that considering both the direction and the magnitude of the keys to compute similarity are important for identifying the tokens to evict. On the other hand, Euclidean shows a significant performance drop relative to KeyDiff.

G.1 Needle in a Haystack results for Sink attention

Table 7: **Math 500 results on DeepSeek-R1-Qwen-7B distilled model (Higher is better)**. We highlight the methods showing the best performance within a given budget with **boldface**.

Max Gen. Length = 4K				Max Gen. Length = 8K			
Num Tokens > 1K (497/500 samples)				Num Tokens > 2K (317/500 samples)			
	Budget	Flex	Exact		Budget	Flex	Exact
Full	N/A	0.762	0.578	Full	N/A	0.816	0.603
KeyDiff	1024	0.664	0.511	KeyDiff	2048	0.713	0.533
SnapKV	1024	**0.690**	0.532	SnapKV	2048	**0.718**	0.533
Num Tokens > 2K (328/500 samples)				Num Tokens > 4K (179/500 samples)			
Full	N/A	0.650	0.478	Full	N/A	0.626	0.441
KeyDiff	2048	0.628	0.464	KeyDiff	4096	0.611	0.419
SnapKV	2048	**0.629**	0.459	SnapKV	4096	**0.615**	0.422

Max Gen. Length = 32K			
Num Tokens > 2K (360/500 samples)			
	Budget	Flex	Exact
Full	N/A	0.889	0.642
KeyDiff	2048	0.713	0.533
SnapKV	2048	**0.718**	0.533
Num Tokens > 4K (141/500 samples)			
Full	N/A	0.787	0.524
KeyDiff	4096	0.611	0.419
SnapKV	4096	**0.615**	0.422
Num Tokens > 8K (53/500 samples)			
Full	N/A	0.550	0.275
KeyDiff	8192	**0.392**	0.222
SnapKV	8192	0.381	0.215

	$\leq 2K$	$2K \leq L \leq 4K$	$4K \leq L \leq 6K$	$6K \leq L \leq 8K$	$\geq 8K$	Total
NarrativeQA	0	0	0	8	192	200
Qasper	1	77	83	25	14	200
MultifidelityQA-En	9	31	21	32	57	150
MultifidelityQA-Zh	14	69	47	33	37	200
HotPotQA	1	4	12	12	171	200
2wikimqa	8	17	68	54	53	200
musique	0	0	0	3	197	200
dureader	0	0	0	16	184	200
gov report	0	20	29	45	106	200
qmsum	0	1	17	15	167	200
multi news	99	71	19	6	5	200
vcsum	4	20	18	32	126	200
trec	4	43	41	39	73	200
triviaqa	4	21	15	21	139	200
samsum	6	29	34	17	114	200
lsht	0	0	3	8	189	200
passage count	0	0	3	13	184	200
passage-retrieval-En	0	0	0	0	200	200
passage-retrieval-Zh	0	0	160	10	0	170
lcc	80	86	21	4	9	200
repobench-p	0	25	37	25	113	200

Table 8: **Distribution of sample length measured by Llama2 tokenizer**

	2K	4K	6K	8K
NarrativeQA	0.10	0.20	0.30	0.40
Qasper	0.47	0.82	0.96	0.98
MultifidelityQA-En	0.40	0.66	0.82	0.93
MultifidelityQA-Zh	0.49	0.78	0.91	0.98
HotPotQA	0.19	0.37	0.52	0.66
2wikimqa	0.36	0.65	0.85	0.92
musique	0.13	0.27	0.40	0.53
dureader	0.17	0.34	0.52	0.68
gov report	0.28	0.52	0.70	0.81
qmsum	0.18	0.36	0.52	0.66
multi news	0.81	0.96	0.98	0.53
vcsum	0.27	0.49	0.65	0.68
trec	0.38	0.66	0.84	0.94
triviaqa	0.26	0.46	0.60	0.72
samsum	0.32	0.55	0.71	0.82
lsht	0.13	0.27	0.39	0.52
passage count	0.15	0.31	0.46	0.60
passage-retrieval-En	0.16	0.33	0.49	0.66
passage-retrieval-Zh	0.37	0.74	0.99	1.00
lcc	0.78	0.95	0.98	0.99
repobench-p	0.27	0.52	0.67	0.78
average	0.31	0.53	0.67	0.77

Table 9: **Compression ratio of prompts w.r.t. various KV cache budgets**

H Additional Experiments and Analyses

To further substantiate the empirical findings presented in Sections 3 and 4, we provide an extended set of experiments and complementary analyses. These additional studies examine the correlation between key geometry and attention, evaluate retrieval-critical and reasoning scenarios, and measure the efficiency of KeyDiff under constrained hardware settings. Together, they offer a broader view of the method's robustness, efficiency, and general applicability across diverse inference conditions.

H.1 Retrieval-Critical Evaluation: Phonebook Lookup

We evaluate KeyDiff on a retrieval-critical setting using the *Phonebook Lookup* task, where the model retrieves a phone number corresponding to a queried name from a long list of entries. Accuracy is averaged across five random phonebooks of varying lengths. As shown in Table 17, KeyDiff maintains high retrieval accuracy for shorter contexts and degrades more gracefully than attention-based baselines as context length increases.

H.2 RULER Benchmark Validation

To assess the generalizability of KeyDiff across diverse architectures, we reference the results reported by the community-maintained KVPress RULER benchmark. KeyDiff consistently achieves competitive or superior leaderboard scores compared to methods such as TOVA, SnapKV, QFilter, and Knorm on both Llama-3.2-3B and Qwen-3-8B backbones, demonstrating its robustness and transferability.

H.3 Needle-in-a-Haystack Recall Saturation

We further analyze the recall behavior of KeyDiff under varying context lengths in the *Needle-in-a-Haystack* benchmark. Table 18 reports the recall difference between KeyDiff and the full-cache baseline as a function of depth and context length. As shown in Table 18, KeyDiff achieves near-parity recall with the full-cache baseline up to 30k tokens, confirming its stability under long-context compression.

H.4 On-Device Latency Evaluation

We further evaluate the runtime characteristics of KEYDIFF on a mobile-class device by measuring the latency required to compute key-eviction scores on an Android platform. All methods are tested on a recent Samsung smartphone under identical FP16 inference precision, and the results are normalized by the latency of KEYDIFF with a cache budget of 512 KVs.

As shown in Table 19, KEYDIFF performs on par with competing methods for small cache sizes and achieves substantially lower scoring latency as the cache size increases, demonstrating both scalability and minimal overhead on edge hardware. Minor runtime fluctuations can be attributed to kernel-level optimizations and proprietary hardware characteristics.

Table 10: Resource unrestricted LongBench Results (Higher is better). All methods processes the input prompt in parallel (i.e., block size = ∞) and make an eviction decision with all token information in the input. The token eviction is made at every step of generation if the budget exceeds. We highlight the methods showing the best performance within a given budget with **boldface**. We omit NarrativeQA from evaluation due to higher chance of OOM errors.

		Single Doc. QA				Multi Doc. QA				Summarization				Fewshot Learning				Synthetic		Code		
		Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	DuReader	GovReport	QMSum	MultiNews	VCSum	TREC	TriviaQA	SAMSum	LSHT	PCount	PR-en	PR-zh	Lcc	RB-P	Avg.
Llama3.1-8B		47.00	56.12	59.86	57.33	47.81	32.25	35.64	34.86	25.32	27.02	17.28	73.00	91.61	43.37	45.50	8.33	99.50	99.00	61.66	51.94	50.72
H2O	2K	22.75	32.73	25.93	43.56	29.49	0.00	5.35	3.70	4.73	17.42	4.44	46.19	54.88	10.39	12.20	16.13	100.00	37.75	42.44	16.44	26.33
	4K	35.49	43.14	39.80	52.82	36.60	10.00	6.11	9.34	7.54	24.61	7.38	54.31	65.83	23.18	16.67	16.39	100.00	61.50	55.96	28.68	34.77
	6K	44.05	47.74	47.88	52.67	45.90	11.11	7.97	15.55	13.77	26.58	9.13	60.11	78.42	32.35	13.64	12.04	100.00	94.00	60.23	39.82	40.65
	8K	45.93	51.12	55.72	53.09	48.83	13.58	15.63	25.89	17.39	27.18	12.39	67.71	86.91	40.05	13.33	13.73	100.00	99.00	60.82	45.17	44.67
TOVA	2K	44.37	55.47	58.07	59.16	48.26	16.67	26.58	30.54	24.37	26.81	16.66	71.00	91.93	45.29	28.89	9.84	100.00	96.58	61.65	51.83	**48.20**
	4K	46.45	56.23	58.82	60.72	49.96	15.28	29.93	33.52	25.56	27.18	17.40	72.08	91.40	44.49	29.79	13.06	100.00	99.00	61.95	52.19	**49.25**
	6K	46.66	54.26	59.31	55.99	50.26	16.22	34.54	34.35	25.21	27.22	17.34	72.96	91.56	44.82	30.43	14.72	100.00	99.00	62.18	53.33	49.52
	8K	46.57	55.44	59.16	59.87	51.77	13.58	35.28	35.21	25.98	27.27	16.80	73.23	90.70	44.71	34.78	13.59	100.00	99.00	61.96	54.32	**49.96**
Sink	2K	33.33	33.74	34.73	45.37	38.46	20.97	18.31	26.08	21.41	24.98	16.08	67.50	90.00	40.99	21.25	2.50	36.00	18.00	57.08	53.81	35.03
	4K	38.57	41.15	46.38	48.07	40.87	22.51	17.50	29.28	21.89	25.12	16.84	71.50	90.52	41.32	24.75	2.50	49.00	22.50	57.94	53.64	38.09
	6K	40.41	43.74	52.60	50.08	42.57	22.32	17.71	30.54	22.38	25.25	17.37	72.50	90.77	41.94	26.25	2.50	57.00	21.00	57.14	54.07	39.41
	8K	40.53	44.58	54.77	49.13	42.38	23.75	20.35	31.35	22.61	25.28	17.45	72.50	90.77	42.27	27.75	3.00	70.00	20.50	57.13	55.35	40.57
SnapKV	2K	45.01	52.53	56.15	57.54	50.17	25.00	32.35	32.99	25.38	27.24	18.14	70.85	89.48	39.71	26.53	15.86	98.57	87.04	60.56	51.92	48.15
	4K	46.28	55.05	59.57	55.49	49.70	24.69	35.09	33.78	26.63	27.15	17.48	72.68	90.28	42.59	28.57	13.38	98.08	98.50	61.14	52.05	49.41
	6K	46.82	55.83	59.07	59.68	50.27	22.22	35.82	34.89	25.17	27.24	17.51	72.31	90.76	44.31	23.26	14.00	100.00	99.00	62.13	54.43	**49.74**
	8K	46.72	55.33	59.79	57.18	51.51	15.28	34.83	35.28	25.71	27.24	17.40	71.96	90.46	45.08	23.26	10.92	100.00	98.99	61.76	54.80	49.17
KeyDiff	2K	44.58	53.88	53.65	57.40	47.66	14.89	33.44	29.97	25.88	26.95	16.10	72.00	92.27	43.51	31.82	11.38	100.00	95.92	59.26	45.66	47.81
	4K	46.05	54.87	57.52	59.04	49.97	13.58	36.51	33.17	25.92	27.12	16.85	73.00	90.13	43.87	31.82	13.28	100.00	97.67	60.57	50.49	49.07
	6K	46.61	54.49	59.19	59.70	50.03	12.99	36.04	34.42	26.67	27.27	17.47	73.33	90.94	44.62	33.33	11.83	100.00	99.00	61.29	53.20	49.62
	8K	46.55	55.11	59.28	57.48	50.51	13.58	35.01	34.81	25.87	27.22	17.15	72.59	91.22	44.69	27.27	11.20	100.00	99.00	61.95	53.62	49.21
Llama3.2-3B		40.23	50.09	55.84	50.69	42.29	26.84	36.24	33.09	24.30	25.21	16.41	72.50	90.11	42.58	34.00	3.00	96.50	20.50	56.22	56.52	43.66
H2O	2K	20.21	24.23	18.98	28.28	23.51	9.88	10.40	1.43	4.13	16.14	3.05	48.00	41.36	12.23	16.00	4.81	1.52	0.50	40.16	17.58	17.12
	4K	32.58	33.87	35.46	35.29	27.46	17.84	12.25	6.96	9.26	23.10	5.17	56.00	59.11	21.77	13.70	4.95	13.71	7.25	51.49	29.58	24.84
	6K	38.61	44.28	46.68	42.34	36.93	11.61	15.57	13.52	13.03	24.44	7.74	63.50	72.81	30.46	12.86	4.76	63.00	19.00	54.28	39.85	32.76
	8K	40.23	44.83	52.90	46.13	39.78	14.74	20.79	22.65	17.06	24.77	10.55	70.50	83.55	35.08	15.07	4.04	81.91	20.00	55.35	45.83	37.29
TOVA	2K	38.22	48.83	54.09	48.08	42.21	14.81	27.89	28.71	23.27	24.94	15.64	70.50	89.47	42.97	22.22	5.88	95.92	18.50	56.14	55.68	**41.20**
	4K	40.55	50.77	56.10	54.26	42.68	17.08	31.66	31.09	23.44	25.37	16.00	71.50	89.77	43.00	21.92	5.88	95.50	18.50	56.33	56.52	42.40
	6K	40.57	50.12	56.62	52.67	43.12	19.44	34.78	32.91	23.88	25.30	15.87	72.50	89.07	42.73	24.00	4.35	94.97	20.00	56.30	57.38	**42.83**
	8K	40.86	49.79	56.31	52.91	42.03	15.34	36.74	33.34	24.11	25.30	16.15	72.50	89.40	42.39	24.32	5.77	95.50	20.00	56.27	56.85	42.79
Sink	2K	33.33	33.74	34.73	45.37	38.46	20.97	18.31	26.08	21.41	24.98	16.08	67.50	90.00	40.99	21.25	2.50	36.00	18.00	57.08	53.81	35.03
	4K	38.57	41.15	46.38	48.07	40.87	22.51	17.50	29.28	21.89	25.12	16.84	71.50	90.52	41.32	24.75	2.50	49.00	22.50	57.94	53.64	38.09
	6K	40.41	43.74	52.60	50.08	42.57	22.32	17.71	30.54	22.38	25.25	17.37	72.50	90.77	41.94	26.25	2.50	57.00	21.00	57.14	54.07	39.41
	8K	40.53	44.58	54.77	49.13	42.38	23.75	20.35	31.35	22.61	25.28	17.45	72.50	90.77	42.27	27.75	3.00	70.00	20.50	57.13	55.35	40.57
KeyDiff	2K	38.15	49.58	51.15	48.73	42.04	20.24	33.47	29.13	23.91	25.10	14.47	70.50	88.31	42.28	20.55	4.90	89.00	17.50	55.21	48.73	40.65
	4K	40.55	51.38	55.88	51.56	41.88	18.78	35.93	31.63	24.15	25.38	15.80	71.50	89.98	42.55	19.18	4.95	95.85	21.00	55.79	55.55	**42.46**
	6K	40.54	51.04	55.42	51.78	41.71	15.26	36.60	32.80	24.67	25.34	16.32	72.50	90.58	43.19	22.22	5.21	97.42	20.50	55.87	56.28	42.76
	8K	40.66	50.93	56.16	53.52	42.07	17.79	37.08	33.31	24.27	25.31	16.00	72.50	89.48	42.52	25.33	5.00	96.46	20.00	56.20	56.80	**43.07**

Table 11: **Full Llama-3.1-8B/3.2-3B-Instruct LongBench Results with $B = 128$ (Higher is better)**. We highlight the methods showing the best performance within a given budget with **boldface**. †: A subset of samples were evaluated due to OOM errors (183/200 samples are evaluated).

		Single Doc. QA				Multi Doc. QA				Summarization				Fewshot Learning				Synthetic			Code		
		Narrative QA	Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	DuReader	GovReport	QMSum	MultiNews	VCSum	TREC	TriviaQA	SAMSum	LSHT	PCount	PR-en	PR-zh	Lcc	RB-P	Avg.
Llama3.1-8B		30.05†	47.00	56.12	59.86	57.33	47.81	32.25	35.64	34.86	25.32	27.02	17.28	73.00	91.61	43.37	45.50	8.33	99.50	99.00	61.66	51.94	49.74
H2O	2K	1.74	21.15	25.33	21.65	26.11	24.15	8.78	5.90	2.17	2.70	16.78	3.97	44.00	29.36	7.62	14.50	2.25	5.88	4.00	40.15	12.14	15.25
	4K	4.07	36.16	36.00	38.02	33.52	32.87	17.78	5.68	6.66	5.95	24.09	6.03	55.00	47.65	17.41	18.50	4.00	24.50	31.25	54.85	21.43	24.83
	6K	8.52	43.31	44.80	46.24	40.03	42.46	21.68	7.33	11.85	8.78	26.03	7.82	62.00	56.39	25.72	18.00	5.75	45.50	90.00	58.62	29.53	33.35
	8K	13.85	44.94	47.81	56.14	43.64	44.90	23.65	11.01	18.78	11.35	26.49	9.96	69.50	69.05	33.41	19.50	5.25	62.50	98.67	59.74	36.26	38.40
TOVA	2K	22.57	37.26	39.43	36.96	45.74	34.48	14.77	16.98	28.87	21.17	26.95	16.21	62.50	90.73	42.74	18.75	0.00	18.00	32.00	62.68	52.48	34.35
	4K	22.68	44.55	47.87	51.16	46.76	44.54	20.56	22.50	30.95	22.13	26.96	16.75	61.50	90.56	43.27	25.25	3.00	43.50	84.00	61.62	53.40	41.12
	6K	24.59	45.93	53.92	55.45	55.09	47.43	25.07	27.68	32.33	24.10	27.00	16.91	68.50	90.81	43.89	29.00	4.25	67.00	98.67	61.50	52.39	45.31
	8K	24.86	46.78	54.83	57.95	54.52	49.00	26.40	31.15	33.44	24.76	27.00	17.33	71.00	91.11	43.29	33.25	6.25	87.00	98.67	61.49	51.79	47.23
Sink	2K	21.83	34.27	29.24	32.82	38.64	29.50	12.59	16.18	28.51	20.21	26.62	15.54	65.00	89.46	42.20	22.25	2.00	25.50	32.50	64.95	59.54	33.78
	4K	22.94	43.01	39.08	46.16	44.04	41.39	19.09	16.54	31.08	21.57	26.78	16.73	70.00	91.53	42.29	29.25	3.00	38.50	71.00	62.12	58.84	39.76
	6K	25.41	47.40	44.13	52.78	47.39	45.73	21.90	17.55	32.53	22.19	26.87	17.05	72.00	91.25	43.41	33.75	3.08	52.50	98.00	62.22	56.24	43.49
	8K	23.53	46.63	48.68	55.77	49.61	47.16	21.14	19.54	33.10	23.20	26.92	16.91	72.00	91.29	43.79	37.00	3.25	66.00	99.00	62.18	56.43	44.91
SnapKV	2K	21.81	37.22	37.19	38.29	46.10	35.42	16.53	16.37	29.83	21.05	26.77	16.16	61.00	88.84	42.56	21.75	4.03	51.50	81.17	62.37	51.45	38.45
	4K	24.79	44.22	47.30	50.27	48.49	46.73	20.55	22.04	32.19	22.68	26.95	16.95	67.50	90.98	43.14	25.00	5.17	89.50	96.67	61.44	51.20	44.46
	6K	24.10	45.57	50.44	55.27	53.12	48.41	24.27	27.46	33.43	23.53	27.03	16.84	71.50	92.28	43.58	27.00	5.25	98.00	99.00	61.32	52.16	46.65
	8K	25.15	46.55	53.39	57.65	56.00	48.75	27.82	32.66	33.67	24.85	27.01	17.37	72.50	91.78	43.54	33.75	5.08	100.00	98.67	61.48	51.41	48.05
KeyL2Norm[9]	2K	8.66	36.63	41.70	37.70	33.75	32.25	5.39	17.73	19.64	14.96	26.69	11.02	63.00	58.94	28.45	22.50	3.05	17.75	20.13	52.40	25.63	27.52
	4K	15.38	44.06	51.75	47.52	50.22	45.56	18.44	27.61	29.50	22.27	26.93	13.44	69.50	79.41	37.50	27.00	4.50	58.00	80.50	58.82	35.08	40.14
	6K	21.75	45.63	55.06	53.77	52.93	47.70	25.63	32.17	32.66	24.85	26.94	15.12	70.00	86.89	40.51	34.50	5.25	75.00	98.00	60.98	43.14	45.17
	8K	25.12	45.70	56.02	56.57	58.14	47.77	30.29	34.09	33.81	24.89	26.94	15.89	71.50	89.26	41.34	39.00	7.25	87.00	99.00	62.05	48.28	47.61
KeyDiff	2K	26.64	41.73	50.99	51.18	51.59	46.47	22.84	32.37	29.02	23.86	26.76	14.81	66.50	85.92	39.26	42.25	3.17	96.00	96.25	59.17	39.42	**45.06**
	4K	28.70	45.62	56.06	56.83	54.58	49.31	28.25	33.06	32.30	25.03	27.07	16.32	70.00	90.85	42.84	44.50	4.21	99.00	97.67	60.80	48.00	**48.14**
	6K	29.90	46.33	55.11	59.00	56.80	49.50	31.52	34.97	33.44	24.58	26.98	16.80	72.00	90.99	43.10	47.00	5.27	99.50	99.00	61.40	49.70	**49.19**
	8K	33.57	46.77	55.48	59.16	56.87	49.37	30.88	34.54	34.17	25.12	27.01	17.13	72.50	92.28	42.81	46.50	5.83	99.50	98.67	61.48	50.90	**49.55**
Llama3.2-3B		23.76	40.23	50.09	55.84	50.69	42.29	26.84	36.24	33.09	24.30	25.21	16.41	72.50	90.11	42.58	34.00	3.00	96.50	20.50	56.22	56.52	42.71
H2O	2K	1.63	19.96	20.20	15.20	18.02	19.56	2.88	6.47	0.78	1.55	15.97	3.11	41.00	21.97	9.83	11.75	0.50	0.50	0.00	39.71	13.91	12.60
	4K	2.92	31.94	33.23	33.25	24.49	28.08	7.55	10.10	5.44	6.30	22.77	4.81	53.00	38.85	20.33	15.50	1.50	7.50	6.25	51.23	22.94	20.38
	6K	4.62	38.81	39.06	45.17	34.66	35.52	15.21	13.36	10.51	10.01	24.25	6.66	61.50	53.23	27.37	15.25	0.50	13.00	19.50	54.55	32.29	26.43
	8K	9.65	39.66	43.20	52.60	38.09	40.41	21.46	18.55	17.80	13.28	24.67	9.12	70.00	64.30	32.19	17.00	2.00	24.50	21.50	55.00	39.09	31.15
TOVA	2K	17.14	30.14	32.44	31.64	35.96	30.05	13.08	9.62	26.15	19.70	25.04	15.47	56.50	87.81	40.48	16.75	2.50	11.50	6.50	55.51	52.36	29.35
	4K	20.52	39.53	42.47	45.80	44.12	38.42	18.22	17.76	29.36	21.36	24.96	16.60	63.50	88.98	41.50	18.75	3.00	23.50	15.00	55.72	56.66	34.56
	6K	20.22	39.78	45.86	52.93	49.08	41.54	20.43	24.78	30.50	22.17	25.11	16.37	66.50	89.00	42.50	21.00	4.00	46.50	20.50	55.57	57.53	37.71
	8K	21.08	40.67	49.07	55.17	48.69	41.93	23.05	31.02	31.64	22.85	25.21	16.55	69.00	89.25	42.19	22.50	2.50	71.00	21.50	55.77	57.47	39.91
Sink	2K	16.85	30.69	26.58	27.32	33.26	25.27	13.82	9.38	26.74	19.15	25.15	15.88	65.00	86.17	40.79	19.50	1.50	19.50	8.50	56.65	52.73	29.54
	4K	19.46	38.61	36.22	41.68	41.97	35.84	13.37	9.86	29.34	20.19	25.06	16.44	71.00	88.06	41.31	21.75	2.50	35.50	16.00	56.48	52.43	33.96
	6K	19.33	40.29	37.95	49.68	46.48	40.29	15.31	11.10	30.43	21.35	25.14	16.64	71.50	88.93	42.04	23.50	3.50	47.00	19.50	56.55	54.11	36.22
	8K	20.15	40.02	41.94	53.57	48.15	42.24	16.01	14.76	31.64	22.10	25.20	16.50	73.00	89.26	42.37	26.25	3.50	62.50	20.50	56.86	56.63	38.25
SnapKV	2K	17.38	31.37	31.48	29.65	37.77	30.05	11.54	9.66	27.03	19.93	24.97	15.97	59.00	88.13	40.48	16.25	3.50	32.50	9.00	56.32	55.91	30.85
	4K	19.85	39.22	39.86	47.33	46.70	37.98	16.64	16.88	29.79	21.21	25.01	16.74	65.50	89.35	40.95	18.25	2.50	62.50	22.50	55.74	56.88	36.73
	6K	20.83	39.65	44.48	51.84	49.30	40.18	20.28	25.32	31.27	22.73	25.09	16.81	69.00	89.95	41.47	18.75	4.00	85.00	20.50	55.69	57.82	39.52
	8K	20.49	40.80	48.16	55.44	48.78	41.65	24.79	30.40	31.81	23.46	25.17	16.44	70.00	90.17	41.99	22.00	5.00	94.00	21.50	55.77	57.29	41.20
KeyL2Norm [9]	2K	7.67	30.39	31.85	30.64	29.47	25.76	7.41	14.17	15.36	12.42	24.20	7.91	48.00	50.99	23.09	17.50	2.00	7.50	5.00	48.92	26.32	22.22
	4K	12.92	37.59	44.71	43.85	38.89	33.42	12.41	22.42	24.63	19.27	24.77	11.37	63.00	72.51	31.75	19.00	3.87	9.50	11.00	55.82	40.08	30.13
	6K	13.02	40.55	48.17	50.87	43.18	40.17	17.10	31.29	28.99	21.47	25.08	13.30	65.00	79.61	37.16	21.50	2.50	46.50	18.50	56.49	47.05	35.60
	8K	15.72	40.54	47.88	54.29	49.29	43.79	22.22	33.18	31.86	22.50	25.19	14.28	70.00	84.92	39.45	23.00	1.50	69.00	20.50	56.82	50.73	38.89
KeyDiff	2K	18.29	36.65	45.44	47.47	46.09	35.41	13.79	28.89	28.16	21.45	25.01	13.56	60.00	85.24	37.00	24.88	1.00	60.50	12.00	54.13	42.01	**35.09**
	4K	22.34	40.60	49.15	52.56	50.14	40.30	21.65	32.46	31.38	23.44	25.06	15.28	66.50	87.92	41.41	27.50	2.50	88.50	19.50	55.55	52.24	**40.28**
	6K	22.29	40.68	50.14	54.51	51.74	42.19	24.83	34.64	32.39	23.53	25.19	15.88	71.00	90.02	42.00	28.75	3.00	95.00	21.50	55.86	54.39	**41.88**
	8K	22.41	40.77	50.10	55.62	49.83	43.58	28.09	34.30	32.78	23.60	25.17	15.77	72.00	90.17	42.46	30.75	3.50	96.50	21.50	55.85	55.65	**42.40**

Table 12: **Full Qwen-2.5-7B/3B-Instruct LongBench Results with** $B = 128$ **(Higher is better)**. We highlight the best and second best methods within a given budget with **bold** and <u>underline</u>.

		Single Doc. QA				Multi Doc. QA				Summarization				Fewshot Learning				Synthetic			Code		
		Narrative QA	Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	DuReader	GovReport	QMSum	MultiNews	VCSum	TREC	TriviaQA	SAMSum	LSHT	PCount	PR-en	PR-zh	Lcc	RB-P	Avg.
Qwen2.5-7B		15.75	16.94	32.38	14.87	11.89	11.88	7.95	30.56	34.33	19.91	22.67	15.28	65.50	87.05	44.75	39.47	4.22	93.08	68.79	57.74	61.84	36.04
H2O	2K	2.39	7.29	12.42	11.73	8.55	11.06	2.73	6.07	3.62	6.60	15.69	3.44	42.50	28.21	10.63	16.00	0.65	0.00	1.50	35.10	18.77	11.66
	4K	1.99	11.92	19.88	14.72	10.24	10.12	4.73	7.51	9.08	10.14	20.85	6.15	51.00	37.37	20.57	15.75	3.16	6.43	27.62	52.14	29.09	17.64
	6K	3.34	14.79	23.94	15.33	11.45	11.30	5.52	9.30	14.63	14.27	22.06	8.68	55.75	51.99	28.01	18.50	1.39	9.41	54.53	54.68	38.32	22.25
	8K	6.10	15.55	28.29	14.99	12.37	14.65	6.24	16.10	20.78	17.22	22.44	11.12	59.00	58.74	33.05	24.92	1.82	15.73	55.16	55.63	44.56	25.45
TOVA	2K	8.49	14.01	21.04	11.55	14.00	11.51	5.09	14.45	27.43	17.84	22.83	15.75	56.50	79.56	40.55	20.50	2.43	9.29	20.45	55.99	56.15	<u>25.02</u>
	4K	12.83	17.03	27.01	14.14	16.80	13.37	8.05	21.15	29.21	19.05	22.73	15.81	58.50	82.67	42.71	27.75	1.67	15.00	43.53	56.69	56.59	28.68
	6K	15.77	15.33	30.31	14.58	19.30	13.78	9.11	25.70	30.40	19.95	22.91	15.10	61.50	83.47	42.90	27.60	1.15	21.75	55.16	57.68	57.99	30.54
	8K	15.69	15.55	33.09	14.78	18.37	13.99	11.26	27.92	31.33	20.17	22.82	15.27	62.00	84.49	43.01	33.21	2.78	30.33	55.16	57.45	58.96	31.79
Sink	2K	7.68	14.68	19.36	12.98	8.58	9.34	3.97	10.66	27.75	17.96	22.33	14.26	62.00	75.26	42.76	23.00	1.07	7.50	21.70	50.11	49.57	23.93
	4K	7.68	17.18	23.46	14.65	9.09	9.38	4.39	10.57	30.23	18.62	22.79	15.48	64.50	83.39	44.19	29.81	2.74	18.08	64.95	55.23	51.30	28.46
	6K	7.37	16.61	25.73	14.74	11.29	11.27	5.69	11.49	31.47	18.72	22.86	15.62	64.50	84.86	44.47	31.07	3.59	41.48	71.21	55.89	55.99	30.76
	8K	8.22	16.15	28.63	15.52	11.59	11.11	6.44	17.29	32.56	18.49	22.91	15.45	65.00	83.95	44.15	35.75	4.14	48.72	71.71	56.82	56.42	31.95
SnapKV	2K	11.60	12.45	23.66	12.54	12.38	10.64	7.03	14.40	27.57	18.27	22.85	15.23	58.00	81.78	41.13	23.67	3.76	19.42	35.09	55.83	56.53	**26.85**
	4K	14.35	13.45	28.28	13.76	16.33	11.74	8.12	21.96	29.71	19.18	22.82	15.20	57.00	83.80	43.27	25.51	2.41	39.83	55.28	58.12	58.67	**30.42**
	6K	14.34	16.35	31.12	14.16	17.56	14.10	8.74	25.56	31.09	20.16	22.84	15.04	60.00	83.80	42.99	30.81	2.91	54.17	55.16	57.48	60.26	32.32
	8K	15.60	15.81	33.47	14.77	18.02	14.49	10.53	27.45	31.99	20.09	22.84	15.20	61.00	84.08	43.01	34.28	4.58	64.25	55.16	57.46	60.59	33.56
KeyDiff	2K	7.17	10.06	24.28	12.96	10.03	10.81	5.71	23.59	17.09	18.03	22.71	11.73	52.00	53.98	32.22	30.00	3.52	33.33	34.37	53.13	32.05	23.75
	4K	13.16	12.00	32.08	14.15	13.04	13.68	5.39	27.83	25.61	20.42	22.76	13.37	54.00	70.90	40.23	39.62	3.37	58.42	54.86	55.95	42.27	<u>30.15</u>
	6K	13.42	14.90	35.11	14.62	18.70	14.09	8.34	30.74	29.83	21.08	22.86	14.39	60.50	77.03	42.00	38.08	4.13	69.83	54.33	56.76	51.50	**32.96**
	8K	14.90	15.77	34.32	14.90	19.02	13.93	9.27	30.65	31.29	20.90	22.79	14.69	60.00	83.01	43.65	40.70	3.87	74.13	55.16	57.33	52.80	**33.96**
Qwen2.5-3B		18.08	22.49	39.72	28.99	27.86	20.45	18.93	32.95	32.80	23.74	24.89	10.95	67.50	85.05	43.88	37.50	5.00	40.97	20.61	51.91	47.53	33.42
H2O	2K	1.80	9.18	11.62	12.62	8.54	7.31	2.70	8.57	5.93	6.97	16.89	4.15	38.00	21.87	7.69	16.00	1.00	3.00	2.69	37.36	22.90	11.75
	4K	2.82	17.34	23.27	21.55	10.18	10.47	3.03	11.05	11.06	10.73	22.93	5.77	50.75	34.93	18.03	16.25	4.35	7.32	16.64	47.74	29.42	17.89
	6K	5.52	18.62	27.93	27.26	12.61	15.07	4.26	14.65	14.92	13.89	24.21	7.55	58.00	45.94	24.93	16.00	2.91	9.10	21.32	49.50	34.54	21.37
	8K	6.16	19.84	32.32	29.66	16.01	17.74	4.99	19.42	20.21	16.49	24.54	9.22	64.00	56.10	32.56	20.25	3.13	11.61	21.32	50.61	38.80	24.52
TOVA	2K	11.69	14.94	25.33	19.90	17.29	12.58	5.91	15.34	26.67	21.49	24.78	16.58	51.50	68.80	41.79	17.75	0.23	6.00	8.68	49.79	48.60	24.08
	4K	12.19	18.31	32.56	27.33	20.58	13.80	7.74	21.11	28.82	22.27	24.98	15.82	59.00	80.66	43.05	21.25	1.11	9.56	19.18	49.93	46.74	27.43
	6K	13.62	19.56	34.64	28.72	21.67	16.25	8.47	27.26	30.17	23.10	24.94	14.53	63.50	81.88	42.97	26.25	1.16	10.58	21.32	51.30	47.70	29.03
	8K	14.66	20.93	37.77	29.72	22.57	17.08	9.63	29.10	31.12	23.17	24.83	13.48	67.00	84.11	43.55	28.25	2.06	13.08	21.32	51.32	47.64	30.11
Sink	2K	9.71	13.75	22.11	20.97	11.63	14.67	4.43	11.89	27.39	19.45	24.36	13.00	56.00	58.77	42.37	22.75	2.50	8.75	4.33	48.27	49.72	<u>23.18</u>
	4K	11.46	18.28	30.40	24.02	15.50	14.62	6.97	11.48	30.08	20.12	24.86	13.35	63.00	68.77	43.11	29.50	3.00	11.75	16.75	51.76	50.47	26.63
	6K	13.01	20.03	32.59	27.06	18.62	15.77	9.37	13.35	30.98	20.70	24.97	13.05	66.50	75.39	42.77	30.00	4.00	14.92	20.44	52.32	50.35	28.39
	8K	10.26	21.27	35.15	29.49	24.31	17.60	9.40	17.59	31.81	21.14	24.99	12.07	68.50	79.17	43.32	34.50	1.00	24.00	20.44	51.47	49.38	29.85
SnapKV	2K	11.70	13.91	24.28	20.75	14.80	10.89	7.42	15.09	27.40	21.63	24.64	15.71	54.50	75.35	42.72	22.38	2.50	18.33	19.06	49.65	50.59	**25.87**
	4K	12.98	22.21	31.77	26.57	18.33	14.41	10.83	21.14	29.14	22.38	24.89	15.88	61.00	84.17	42.63	21.17	3.75	25.42	22.46	50.22	48.77	**29.05**
	6K	14.16	20.09	36.15	28.41	19.14	15.59	12.70	26.21	30.35	22.75	24.91	14.96	65.00	83.92	43.52	25.50	5.00	32.20	21.32	51.04	47.49	**30.50**
	8K	12.76	20.88	37.10	30.10	22.49	18.19	13.83	29.54	31.33	23.37	24.80	13.75	65.50	84.88	44.49	28.00	5.20	35.83	21.32	51.31	47.82	31.55
KeyDiff	2K	3.99	10.20	22.71	15.77	8.93	13.12	5.51	24.79	17.35	16.56	24.31	10.53	57.50	41.19	27.43	25.25	3.88	11.32	10.68	46.44	34.33	20.56
	4K	9.39	18.61	31.37	23.64	18.96	18.10	7.86	27.01	25.64	22.28	24.70	10.93	65.00	63.02	37.74	30.50	4.00	20.08	26.85	49.27	39.24	<u>27.34</u>
	6K	10.51	19.71	35.51	28.89	26.92	18.28	11.47	31.83	29.32	23.63	24.90	11.46	64.50	75.36	41.51	34.50	3.57	31.41	21.32	50.60	42.95	<u>30.39</u>
	8K	12.24	20.49	38.52	29.60	23.05	19.41	15.95	30.92	31.10	23.89	24.83	11.80	67.50	79.05	41.73	36.50	3.08	40.21	21.32	51.05	45.88	**31.82**

Table 13: **Llama-3.2-3B-Instruct LongBench Results with prompt block size $B \in [64, 256]$ (Higher is better)**. We highlight the best and second best methods within a given budget with **bold** and <u>underline</u>.

		Single Doc. QA				Multi Doc. QA				Summarization				Fewshot Learning				Synthetic			Code		
		Narrative QA	Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	DuReader	GovReport	QMSum	MultiNews	VCSum	TREC	TriviaQA	SAMSum	LSHT	PCount	PR-en	PR-zh	Lcc	RB-P	Avg.
$B = 64$		23.76	40.23	50.09	55.84	50.69	42.29	26.84	36.24	33.09	24.30	25.21	16.41	72.50	90.11	42.58	34.00	3.00	96.50	20.50	56.22	56.52	42.71
H2O	2K	1.30	18.23	16.96	14.25	12.26	16.84	0.72	6.88	0.78	1.29	16.24	2.97	35.00	18.07	9.78	13.50	0.50	1.50	0.25	39.98	12.68	11.43
	4K	2.30	31.92	31.59	32.44	25.02	22.58	4.89	9.29	5.36	5.57	23.02	4.80	50.50	33.05	18.76	13.75	1.00	2.50	3.75	50.11	21.57	18.75
	6K	3.12	38.87	37.63	44.58	34.38	35.35	12.13	12.82	10.43	9.38	24.31	6.63	63.00	45.46	26.08	14.25	0.00	8.50	17.50	53.84	30.66	25.19
	8K	9.11	40.09	45.04	52.58	39.24	38.25	15.88	17.82	17.97	12.96	24.70	9.26	70.50	60.57	32.71	16.00	0.50	21.50	19.50	54.72	39.26	30.39
TOVA	2K	17.24	30.03	31.04	32.07	36.58	28.97	12.17	10.50	26.35	19.78	25.07	15.20	60.50	87.45	41.07	15.50	1.00	10.50	6.00	55.30	52.36	29.27
	4K	19.59	39.27	42.16	44.54	44.58	37.63	18.62	17.44	28.82	21.46	25.18	16.49	62.50	89.48	41.89	17.50	3.50	24.00	15.00	55.14	56.58	34.35
	6K	21.53	40.32	46.16	52.81	49.44	40.35	18.73	25.74	30.47	22.30	25.23	16.23	66.00	90.00	42.48	21.00	3.00	47.00	18.50	55.15	58.03	37.64
	8K	21.32	40.87	50.20	54.84	49.35	42.11	24.52	30.71	31.60	23.05	25.20	16.57	69.00	90.50	41.80	23.25	5.50	74.50	19.50	55.45	58.34	40.39
Sink	2K	15.68	29.91	26.61	27.42	33.16	25.43	13.36	9.37	26.70	19.25	25.01	16.04	64.50	86.33	41.04	19.00	1.50	19.00	8.50	56.48	52.91	29.39
	4K	19.35	37.77	36.91	41.61	41.46	35.26	12.88	10.30	29.59	20.27	25.01	16.09	69.00	88.06	41.88	21.25	2.50	36.00	16.50	55.85	52.51	33.81
	6K	19.36	40.01	38.16	49.57	46.39	39.01	14.20	10.41	30.54	21.64	25.07	16.93	71.00	88.50	42.09	23.50	3.50	47.50	19.50	56.02	53.97	36.04
	8K	20.49	39.89	42.21	53.08	47.72	41.44	16.37	15.08	31.80	22.04	25.07	16.89	72.00	89.26	42.20	25.25	3.50	61.50	20.50	56.21	56.42	38.04
SnapKV	2K	18.07	31.21	30.60	31.32	37.31	30.69	11.71	9.98	26.98	19.87	25.13	16.05	61.00	87.85	40.36	16.25	3.00	32.00	8.00	56.78	55.77	<u>30.95</u>
	4K	19.30	39.01	40.81	44.86	47.83	37.74	16.75	16.94	29.90	21.37	25.25	16.70	65.00	88.88	40.99	16.75	3.50	59.50	18.00	55.15	57.20	<u>36.26</u>
	6K	20.85	40.32	45.58	52.80	48.03	41.63	18.47	24.98	30.68	22.32	25.12	16.71	68.50	90.00	41.49	18.25	4.50	85.00	18.50	55.46	57.30	<u>39.36</u>
	8K	20.64	41.10	47.89	54.70	48.49	41.79	21.58	31.46	32.03	23.32	25.22	16.65	71.00	90.00	41.44	21.00	4.00	95.00	19.50	55.44	57.54	<u>40.94</u>
KeyDiff	2K	17.40	38.12	45.25	47.09	45.28	34.23	13.97	27.96	28.34	21.09	24.94	13.45	56.00	83.29	38.53	24.25	1.00	63.50	12.50	54.41	40.00	**34.79**
	4K	22.38	42.00	50.84	53.16	47.34	40.56	21.43	32.82	30.96	23.32	25.08	15.35	67.50	87.09	42.53	27.50	2.00	89.00	18.50	54.39	53.09	**40.33**
	6K	22.25	41.55	50.32	54.91	51.61	42.02	24.62	34.60	32.40	23.63	25.26	16.18	71.00	88.42	41.90	29.25	3.50	95.00	19.50	55.69	55.27	**41.85**
	8K	21.57	41.24	50.12	55.33	49.98	43.78	27.45	34.30	32.43	23.67	25.21	15.99	71.50	90.84	42.32	30.00	3.00	96.50	19.50	55.54	56.56	**42.23**
$B = 256$		23.76	40.23	50.09	55.84	50.69	42.29	26.84	36.24	33.09	24.30	25.21	16.41	72.50	90.11	42.58	34.00	3.00	96.50	20.50	56.22	56.52	42.71
H2O	2K	1.87	19.19	23.35	16.06	20.95	17.91	2.33	7.31	0.83	1.73	16.29	3.28	42.50	25.82	9.83	14.50	2.75	0.00	1.00	38.59	14.57	13.36
	4K	5.58	31.49	33.28	32.36	26.85	28.34	8.61	10.83	5.46	6.27	23.13	4.98	54.00	41.73	19.14	15.00	1.00	8.00	6.50	51.39	24.24	20.87
	6K	7.49	37.99	41.54	44.71	39.53	36.18	15.46	13.87	10.77	10.46	24.60	6.94	61.50	58.64	27.20	15.50	1.00	15.00	19.00	54.30	32.45	27.34
	8K	9.92	39.71	43.84	52.15	39.14	39.23	19.28	18.04	17.78	13.60	24.98	8.99	71.00	67.64	32.83	17.25	2.50	30.00	21.00	55.25	39.33	31.59
TOVA	2K	18.46	30.80	33.74	32.24	39.73	32.18	14.10	10.17	26.32	20.17	25.18	15.67	62.00	89.36	40.60	16.25	2.00	16.00	4.50	55.98	53.42	30.42
	4K	20.36	38.18	42.53	46.18	46.83	36.60	17.81	17.09	28.99	20.54	25.23	16.34	63.50	89.13	41.55	19.12	3.50	29.50	14.50	55.55	55.91	34.71
	6K	20.71	40.46	45.82	52.95	51.33	40.92	21.47	24.80	30.71	22.37	25.48	16.40	67.00	88.50	41.91	20.25	3.50	49.00	20.00	55.85	56.74	37.91
	8K	20.84	40.79	48.02	54.82	50.12	40.71	25.17	30.85	31.47	22.98	25.49	16.41	71.00	89.00	41.99	22.25	2.00	76.50	21.00	55.93	57.59	40.23
Sink	2K	15.48	30.74	26.93	27.76	33.86	25.63	13.30	9.56	26.70	19.41	25.15	15.56	66.00	86.44	41.17	18.50	1.00	19.00	8.50	56.47	52.37	29.50
	4K	18.91	38.38	37.28	41.76	41.81	35.17	13.07	9.96	29.23	20.61	25.01	16.38	70.50	88.06	42.07	21.75	1.50	36.00	17.00	56.05	51.98	33.93
	6K	19.37	40.30	38.14	49.77	46.14	39.60	14.18	11.12	30.48	21.54	25.20	16.28	71.00	88.80	41.79	23.50	3.50	47.00	20.00	55.65	53.53	36.04
	8K	20.35	40.19	42.96	53.25	47.82	41.10	17.87	14.83	31.32	22.14	25.17	16.56	72.50	89.26	42.47	26.25	5.00	62.50	20.00	56.00	56.03	38.27
SnapKV	2K	17.04	31.55	31.75	31.87	37.25	34.03	12.17	9.80	27.17	20.16	25.26	16.27	61.00	87.63	40.95	17.50	4.00	35.00	10.00	55.93	54.39	<u>31.46</u>
	4K	19.67	39.34	40.95	45.81	44.27	38.78	16.17	16.61	29.99	21.15	25.49	16.71	66.50	89.15	40.76	19.75	3.50	66.00	19.00	55.44	56.45	<u>36.74</u>
	6K	22.98	40.13	44.80	52.39	50.61	39.28	20.31	24.84	31.28	22.17	25.42	16.55	70.00	89.79	41.91	19.75	3.50	86.50	20.00	55.93	57.49	<u>39.79</u>
	8K	20.23	40.75	47.40	54.77	49.74	41.47	24.74	31.19	31.95	22.96	25.45	16.27	71.50	90.17	42.12	23.25	4.00	93.50	21.00	55.93	57.36	<u>41.23</u>
KeyDiff	2K	18.99	37.20	46.57	46.61	45.14	33.12	15.15	29.54	28.30	21.92	25.41	14.44	58.50	86.30	37.77	23.75	1.00	67.00	13.50	54.07	41.59	**35.52**
	4K	21.00	41.48	48.69	52.94	47.07	40.26	22.69	33.07	31.11	23.35	25.22	15.57	65.50	88.72	41.79	26.50	2.00	91.00	19.00	55.79	51.50	**40.20**
	6K	21.61	40.78	49.68	55.08	50.64	41.38	24.57	34.62	32.02	23.36	25.56	15.86	71.50	89.42	42.37	28.75	3.00	94.00	21.00	55.96	55.96	**41.77**
	8K	22.24	40.83	50.23	55.45	50.50	43.28	28.37	33.88	32.67	23.85	25.48	15.83	72.50	90.34	42.31	30.50	3.00	95.50	21.00	56.01	55.58	**42.35**

Table 14: **KEYDIFF + Recent tokens Llama-3.1-8B/3.2-3B-Instruct LongBench Results with** $B = 128$ **(Higher is better)**. We highlight the methods showing the best performance within a given budget with **boldface**. X% indicates X% of KV cache budget is reserved to keep the recent tokens, while the remaining cache budget is managed by KEYDIFF algorithm. †: A subset of samples were evaluated due to OOM errors (183/200 samples are evaluated).

		Single Doc. QA				Multi Doc. QA				Summarization				Fewshot Learning				Synthetic			Code		
		Narrative QA	Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	DuReader	GovReport	QMSum	MultiNews	VCSum	TREC	TriviaQA	SAMSum	LSHT	PCount	PR-en	PR-zh	Lcc	RB-P	Avg.
Llama3.1-8B		30.05†	47.00	56.12	59.86	57.33	47.81	32.25	35.64	34.86	25.32	27.02	17.28	73.00	91.61	43.37	45.50	8.33	99.50	99.00	61.66	51.94	49.74
Sink	2K	21.83	34.27	29.24	32.82	38.64	29.50	12.59	16.18	28.51	20.21	26.62	15.54	65.00	89.46	42.20	22.25	2.00	25.50	32.50	64.95	59.54	33.78
	4K	22.94	43.01	39.08	46.16	44.04	41.39	19.09	16.54	31.08	21.57	26.78	16.73	70.00	91.53	42.29	29.25	3.00	38.50	71.00	62.12	58.84	39.76
	6K	25.41	47.40	44.13	52.78	47.39	45.73	21.90	17.55	32.53	22.19	26.87	17.05	72.00	91.25	43.41	33.75	3.08	52.50	98.00	62.22	56.24	43.49
	8K	23.53	46.63	48.68	55.77	49.61	47.16	21.14	19.54	33.10	23.20	26.92	16.91	72.00	91.29	43.79	37.00	3.25	66.00	99.00	62.18	56.43	44.91
SnapKV	2K	21.81	37.22	37.19	38.29	46.10	35.42	16.53	16.37	29.83	21.05	26.77	16.16	61.00	88.84	42.56	21.75	4.03	51.50	81.17	62.37	51.45	38.45
	4K	24.79	44.22	47.30	50.27	48.49	46.73	20.55	22.04	32.19	22.68	26.95	16.95	67.50	90.98	43.14	25.00	5.17	89.50	96.67	61.44	51.20	44.46
	6K	24.10	45.57	50.44	55.27	53.12	48.41	24.27	27.46	33.43	23.53	27.03	16.84	71.50	92.28	43.58	27.00	5.25	98.00	99.00	61.32	52.16	46.65
	8K	25.15	46.55	53.39	57.65	56.00	48.75	27.82	32.66	33.67	24.85	27.01	17.37	72.50	91.78	43.54	33.75	5.08	100.00	98.67	61.48	51.41	48.05
KeyDiff + 10% Recent Tokens	2K	27.36	40.79	50.63	50.21	49.44	44.77	25.04	30.25	29.41	23.28	26.76	15.78	64.00	85.90	43.76	44.00	4.25	98.50	95.83	62.97	48.70	**45.79**
	4K	27.42	45.67	53.62	57.40	53.77	47.30	27.48	33.40	31.90	24.18	26.83	16.51	70.00	88.89	43.95	47.00	5.21	99.00	98.67	62.20	52.98	**48.26**
	6K	30.58	46.24	55.58	57.82	56.44	47.74	29.03	34.33	33.15	24.67	26.91	16.98	71.50	92.22	43.96	47.00	4.23	99.50	99.00	62.56	52.98	49.16
	8K	32.17	46.66	55.65	58.65	57.24	48.64	30.54	33.44	33.85	24.93	26.94	17.12	72.50	91.72	43.70	46.00	5.47	99.50	99.00	62.56	51.75	49.43
KeyDiff + 20% Recent Tokens	2K	26.73	41.88	49.79	48.46	49.68	42.51	26.90	30.90	28.83	23.34	26.76	16.11	62.50	87.39	43.99	42.50	4.58	94.00	95.58	63.33	50.01	45.51
	4K	26.05	45.11	55.51	55.87	54.14	47.41	25.52	33.51	31.87	24.40	26.95	16.09	70.00	90.22	43.76	43.50	4.00	99.50	98.67	62.06	52.27	47.92
	6K	28.39	46.43	55.12	58.27	57.02	48.94	28.98	34.31	33.32	24.62	26.96	17.20	71.50	91.72	44.02	46.50	5.13	99.50	99.00	62.39	52.57	49.14
	8K	31.35	46.74	54.71	58.60	58.19	48.14	31.77	34.02	33.55	24.98	26.95	17.11	72.50	91.72	44.17	46.00	7.38	99.50	99.00	62.39	52.04	**49.56**
KEYDIFF	2K	26.64	41.73	50.99	51.18	51.59	46.47	22.84	32.37	29.02	23.86	26.76	14.81	66.50	85.92	39.26	42.25	3.17	96.00	96.25	59.17	39.42	45.06
	4K	28.70	45.62	56.06	56.83	54.58	49.31	28.25	33.06	32.30	25.03	27.07	16.32	70.00	90.85	42.84	44.50	4.21	99.00	97.67	60.80	48.00	48.14
	6K	29.90	46.33	55.11	59.00	56.80	49.50	31.52	34.97	33.44	24.58	26.98	16.80	72.00	90.99	43.10	47.00	5.27	99.50	99.00	61.40	49.70	**49.19**
	8K	33.57	46.77	55.48	59.16	56.87	49.37	30.88	34.54	34.17	25.12	27.01	17.13	72.50	92.28	42.81	46.50	5.83	99.50	98.67	61.48	50.90	49.55
Llama3.2-3B		23.76	40.23	50.09	55.84	50.69	42.29	26.84	36.24	33.09	24.30	25.21	16.41	72.50	90.11	42.58	34.00	3.00	96.50	20.50	56.22	56.52	42.71
Sink	2K	16.85	30.69	26.58	27.32	33.26	25.27	13.82	9.38	26.74	19.15	25.15	15.88	65.00	86.17	40.79	19.50	1.50	19.50	8.50	56.65	52.73	29.54
	4K	19.46	38.61	36.22	41.68	41.97	35.84	13.37	9.86	29.34	20.19	25.06	16.44	71.00	88.06	41.31	21.75	2.50	35.50	16.00	56.48	52.43	33.96
	6K	19.33	40.29	37.95	49.68	46.48	40.29	15.31	11.10	30.43	21.35	25.14	16.64	71.50	88.93	42.04	23.50	3.50	47.00	19.50	56.55	54.11	36.22
	8K	20.15	40.02	41.94	53.57	48.15	42.24	16.01	14.76	31.64	22.10	25.20	16.50	73.00	89.26	42.37	26.25	3.50	62.50	20.50	56.86	56.63	38.25
SnapKV	2K	17.38	31.37	31.48	29.65	37.77	30.05	11.54	9.66	27.03	19.93	24.97	15.97	59.00	88.13	40.48	16.25	3.50	32.50	9.00	56.32	55.91	30.85
	4K	19.85	39.22	39.86	47.33	46.70	37.98	16.64	16.88	29.79	21.21	25.01	16.74	65.50	89.35	40.95	18.25	2.50	62.50	22.50	55.74	56.88	36.73
	6K	20.83	39.65	44.48	51.84	49.30	40.18	20.28	25.32	31.27	22.73	25.09	16.81	69.00	89.95	41.47	18.75	4.00	85.00	20.50	55.69	57.82	39.52
	8K	20.49	40.80	48.16	55.44	48.78	41.65	24.79	30.40	31.81	23.46	25.17	16.44	70.00	90.17	41.99	22.00	5.00	94.00	21.50	55.77	57.29	41.20
KeyDiff + 10% Recent Tokens	2K	19.92	37.58	45.99	46.41	44.24	34.81	15.20	29.22	28.23	22.49	24.99	14.71	59.00	84.08	41.38	24.38	3.00	70.50	15.50	57.49	51.91	**36.72**
	4K	21.97	40.70	49.76	52.53	47.79	41.92	20.71	32.83	31.12	23.37	25.07	15.77	67.50	87.47	41.28	25.00	2.50	91.00	20.00	57.14	56.25	**40.56**
	6K	23.76	40.56	50.79	54.23	50.54	42.15	25.20	34.12	31.95	23.10	25.21	15.91	71.50	88.17	41.67	26.50	3.50	96.00	20.50	56.80	55.18	41.78
	8K	23.65	40.58	49.96	55.71	51.52	44.10	25.95	34.41	32.80	23.77	25.25	15.75	73.50	89.67	41.83	28.75	3.00	96.00	20.50	57.00	56.12	42.37
KeyDiff + 20% Recent Tokens	2K	19.27	34.86	45.26	44.94	42.81	34.15	14.27	27.31	28.10	22.13	25.08	14.96	61.50	85.07	41.62	24.17	2.00	71.50	16.00	57.56	53.58	36.48
	4K	22.56	41.28	48.37	52.00	47.50	42.64	19.64	32.28	31.17	22.83	25.08	15.68	68.50	88.17	41.38	24.25	2.00	90.50	19.50	57.04	55.05	40.35
	6K	22.88	40.74	50.37	55.02	49.90	42.34	25.02	34.95	31.97	23.28	25.24	16.32	71.50	89.61	41.43	27.00	3.00	95.50	20.50	56.82	55.77	41.86
	8K	23.82	40.58	50.02	55.71	51.25	43.64	24.52	34.33	33.02	23.82	25.20	16.08	73.50	89.67	41.99	29.25	3.00	97.00	20.50	57.05	56.48	**42.40**
KEYDIFF	2K	18.29	36.65	45.44	47.47	46.09	35.41	13.79	28.89	28.16	21.45	25.01	13.56	60.00	85.24	37.00	24.88	1.00	60.50	12.00	54.13	42.01	35.09
	4K	22.34	40.60	49.15	52.56	50.14	40.30	21.65	32.46	31.38	23.44	25.06	15.28	66.50	87.92	41.41	27.50	2.50	88.50	19.50	55.55	52.24	40.28
	6K	22.29	40.68	50.14	54.51	51.74	42.19	24.83	34.64	32.39	23.53	25.19	15.88	71.00	90.02	42.00	28.75	3.00	95.00	21.50	55.86	54.39	**41.88**
	8K	22.41	40.77	50.10	55.62	49.83	43.58	28.09	34.30	32.78	23.60	25.17	15.77	72.00	90.17	42.46	30.75	3.50	96.50	21.50	55.85	55.65	42.40

Table 15: **Anchor vector ablation study.** Average of Full Longbench results for Llama 3.2-3B-Instruct. KeyDiff results match Table 11. (Higher is better)

	2K	4K	6K	8K
KeyDiff	35.09	40.28	41.88	42.40
Pairwise	35.24	40.61	41.87	42.45
Median	35.43	40.67	41.89	42.26

Table 16: **Distance metric ablation study.** Average of Full Longbench results of Llama 3.2-3B-Instruct. KeyDiff results match Table 11. (Higher is better)

	2K	4K	6K	8K
KeyDiff	35.09	40.28	41.88	42.40
DotProd	30.14	38.01	41.09	42.23
Euclidean	13.68	21.06	25.91	29.53

Table 17: Accuracy (%) on the Phonebook Lookup task using Llama-3.2-3B-Instruct with a 6k cache budget.

Method	100	478	856	1233	1611	1989	2367	2744	3122	3500
Dense	1.0	1.0	1.0	0.8	1.0	0.8	0.6	0.6	0.8	0.6
KeyDiff	1.0	1.0	1.0	0.6	0.4	0.2	0.0	0.0	0.2	0.2
TOVA	1.0	1.0	1.0	0.4	0.2	0.0	0.0	0.0	0.0	0.0

Table 18: Recall difference (KeyDiff-Full) on NIAH benchmark for Llama-3.2-3B.

Depth	1000	4222	7444	10667	13889	17111	20333	23556	26778	30000
0.0	0.02	−0.06	−0.06	0.20	0.16	−0.16	0.18	0.20	0.06	0.04
11.0	0.10	−0.04	−0.04	0.30	0.28	−0.18	0.26	0.28	0.10	−0.08
22.0	0.04	−0.12	0.00	0.32	0.30	−0.18	0.16	0.28	0.14	−0.04
33.0	0.02	0.06	−0.20	0.34	0.08	−0.16	0.26	0.26	0.20	−0.16
44.0	−0.06	−0.08	−0.16	0.30	0.26	−0.12	0.24	0.22	−0.10	0.04
56.0	−0.06	0.00	−0.24	0.26	0.26	−0.20	0.18	0.14	0.04	−0.02
67.0	0.02	0.02	−0.14	0.36	0.28	−0.26	0.24	0.16	0.04	−0.08
78.0	−0.10	0.08	−0.26	0.26	0.30	−0.26	0.20	0.30	0.02	−0.04
89.0	0.02	0.08	−0.06	0.30	0.26	−0.32	0.30	0.22	0.06	−0.12
100.0	0.00	−0.06	−0.26	0.00	−0.06	−0.02	0.00	−0.04	0.14	−0.04

Table 19: Relative latency of key scoring on an Android device (normalized by KeyDiff at 512 KVs). Lower is better.

Method	512	1024	2048	4096	8192
SnapKV	1.92	2.46	4.50	8.94	11.42
TOVA	1.02	0.94	1.25	1.95	3.06
KeyDiff	**1.00**	1.03	1.01	1.07	1.37
H2O	1.10	0.99	1.52	2.32	4.23

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims made in the abstract are corroborated by both our theoretical justification Section 3.3, experiments Section 4 and auxiliary results and discussion.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]

   Justification: In Section 6 we discuss the fact that KEYDIFF is only tested on decoder based models featuring GQA and would like to extend it in the future to models with different designs such as MLA.

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

   Justification: Proofs of all results are discussed briefly in their statement in Section 3.3 and rigorously proved in Appendix C.3.

4. **Experimental Result Reproducibility**

   Question: If the contribution is a dataset or model, what steps did you take to make your results reproducible or verifiable? Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), release of a model checkpoint, or other means that are appropriate to your research.

   Answer: [NA]

   Justification: The paper releases neither a dataset or a model, though the cache management algorithm can be easily implemented using the description from the paper.

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

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

   Answer: [No]

   Justification: Code is not provided as it is proprietary. However the algorithm may be readily implemented and results replicated using the description in the paper.

6. **Experimental Setting/Details**

   Question: If you ran experiments, did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? The full details can be provided with the code, but the important details should be in the main paper, and information about how hyperparameters were selected should appear either in the paper or supplementary materials.

Answer: [Yes]

Justification: All hyperparameters selected for the benchmarks run to evaluate KEYDIFF are specified in Section 4 and Appendices E and F.2. Hyperparameters to recreate figures are given in figure descriptions.

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: The only experiment using stochasticity Figure 7 and Figure 17 reports error bars and statistical significance. The Math500 results as well discuss the use of sampling and stochasticity in generating responses. Aside from this, the remaining results are deterministic.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: In all relevant cases, particularly experiments on TTFT Figure 17 and Figure 7, we mention the GPUs and compute resources used.

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research proposes a new cache management algorithm based on the geometry of the attention mechanism in decoder-based LLMs. It does not involve human subjects or obviously ethically sensitive applications, and it conforms to the NeurIPS Code of Ethics.

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: This research is foundational in nature and is not tied to particular applications or deployments. As such we do not see any direct path to negative applications of our work.

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper proposes a new cache eviction algorithm. Neither the models used, data or cache eviction algorithm appear to pose a high risk for abuse necessitating release safeguards beyond standard practices.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: Creators for datasets/benchmarks LongBench and Math500 are properly credited with citations. Open source models from the Llama and Qwen model classes are used to test KEYDIFF are properly cited as are their deepseek distill r1 variants.

13. **Assets**

    Question: If you are releasing new assets, did you document them and provide these details alongside the assets? Researchers should communicate the details of the dataset or the model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

    Answer: [NA]

    Justification: The research does not release new assets.

14. **Crowdsourcing and Research with Human Subjects**

    Question: If you used crowdsourcing or conducted research with human subjects, did you include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)? Including this information in the supplemental material is fine, but if the main contribution of your paper involves human subjects, then we strongly encourage you to include as much detail as possible in the main paper. According to the NeurIPS Code of Ethics, you must pay workers involved in data collection, curation, or other labor at least the minimum wage in your country.

    Answer: [NA]

    Justification: The research does not involve crowdsourcing experiments or research with human subjects.

15. **IRB Approvals**

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

    Answer: [NA]

    Justification: The research does not involve human subjects, therefore IRB approval is not applicable.

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: The research does not include any input from an LLM. In particular the core methodology and research is original to the authors.