DELTAKV: BEYOND ATTENTION MAGNITUDE FOR KV CACHE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

The growing memory footprint and computation cost of the key–value (KV) cache pose a longstanding bottleneck for efficient long-context processing. KV eviction mitigates these costs by retaining a subset of cached entries, and most existing methods decide which entries to retain based on attention scores. However, our analysis shows that attention scores exhibit both query-relevant and query-invariant components; attention over the same context exhibits substantial shared patterns across different queries. Under a limited KV budget, these shared patterns may bias KV selection toward entries that are less relevant to the current query. Motivated by this observation, we propose DELTAKV, a KV eviction method that scores cached entries by contrasting attention under the actual query with attention under a generic proxy query. This simple contrast discounts shared structure and emphasizes query-specific changes to guide KV selection, with minimal computational overhead. Experiments on LongBench-E, RULER, and MileBench show that DELTAKV degrades more gracefully than competing methods as the compression ratio increases. Across these three benchmarks, evicting 90% of KV entries costs less than 7% relative performance. Even with only 2% of the original KV cache, DeltaKV retains over 70% of full-cache performance and outperforms all evaluated baselines in every model–benchmark combination.

1 INTRODUCTION

Long-context processing is increasingly important for large language models (LLMs) and multimodal LLMs in applications such as document question answering, multi-session dialogue, and video understanding (Bai et al., 2024; Wu et al., 2024; Fu et al., 2025). These applications require models to integrate information distributed across lengthy inputs and maintain access to it throughout generation. Supporting such workloads efficiently places growing demands on the key–value (KV) cache, which stores contextual representations for reuse during autoregressive decoding (Liu et al., 2024a). Its memory footprint grows linearly with the number of cached tokens, while longer contexts increase the amount of cached data accessed at each decoding step. KV eviction reduces these memory and bandwidth costs by retaining only a subset of entries. The central challenge is deciding which entries to preserve so that a smaller cache retains the information needed for accurate generation.

Many existing KV-cache compression methods use attention magnitude as a proxy for the importance of cached entries. The attention signals used for selection range from attention over the context in SnapKV (Li et al., 2024a) and KVzip (Kim et al., 2026a) to text-to-visual attention in VL-Cache (Tu et al., 2025). Attention distributions also inform cache-budget allocation across heads and layers (Feng et al., 2026; Wan et al., 2025) and guide cross-modal KV merging (Li et al., 2025). These approaches reflect the widespread use of attention to determine which contextual information to preserve, with attention-based entry selection motivated by the intuition that entries receiving higher attention are more useful for generation.

In this work, we show that prevailing attention-based KV-cache compression strategies can be suboptimal and investigate the attention behavior underlying this limitation. We identify substantial shared patterns that persist across queries and coexist with query-specific attention. This indicates that high attention does not necessarily imply relevance to the current question. Under a limited KV budget, these shared patterns may cause entries that consistently attract attention across queries

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to receive higher priority than information needed to answer the current question. These findings motivate distinguishing query-specific attention from shared patterns when determining which contextual information to preserve.

Motivated by these observations, we introduce **DeltaKV**, a training-free KV eviction method that compares attention against a reference pattern to attenuate shared structure and highlight query-specific relevance. To construct this reference, we use a *proxy query* to estimate the attention patterns shared across queries over the same cached context. We then compute each entry's importance by subtracting its proxy-query attention score from its attention score under the *real query*, the task query for which an answer is generated. This subtraction downweights entries that receive similarly high attention under both queries and emphasizes entries whose attention increases specifically for the real query. DeltaKV uses the resulting scores to select context KV entries before decoding. Evaluations against existing KV eviction methods on LongBench-E (Bai et al., 2024), RULER (Hsieh et al., 2024), MileBench (Song et al., 2024), and Video-MME (Fu et al., 2025) demonstrate robust performance under aggressive compression across text and multimodal tasks: DeltaKV retains over 93% of full-cache performance on all four benchmarks while preserving only 10% of KV entries. Additional experiments assess robustness to design choices, compatibility with INT4 KV-cache quantization, and latency and memory efficiency.

Our main contributions are:

- We identify attention patterns shared across different queries and analyze their implications for KV retention under limited memory budgets.
- We propose DeltaKV, a training-free KV eviction method that contrasts real and proxy query attention to attenuate shared patterns and emphasize query-specific relevance.
- We demonstrate robust performance under aggressive compression across modalities and model sizes, and evaluate memory savings, latency trade-offs, and compatibility with INT4 KV-cache quantization.

2 Related Work

Attention and KV caching. Transformers aggregate contextual information through scaled dot-product attention (Vaswani et al., 2017):

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M\right) \tag{1}$$

where Q and K denote query and key matrices, d is the key dimension, and M is the causal mask. The attention weight A_{ij} determines the contribution of value vector v_j to the output for query q_i.

During autoregressive inference, prefill constructs the KV cache from the input prompt, while decoding reuses and extends it one token at a time. Caching avoids recomputation but incurs substantial memory overhead for long contexts. Multi-query attention (MQA) (Shazeer, 2019) and grouped-query attention (GQA) (Ainslie et al., 2023) reduce per-token storage by sharing KV heads across query heads, while multi-head latent attention (MLA) (Liu et al., 2024a) caches a compact latent representation through joint low-rank KV compression. Despite these reductions, cache memory still grows linearly with sequence length, motivating further KV-cache compression.

Attention-based KV cache compression. Many methods use attention statistics as a proxy for KV-entry importance across text and multimodal models. Importance scores are derived from accumulated attention history (Zhang et al., 2023; Liu et al., 2023) or attention within designated observation windows (Li et al., 2024a; Tu et al., 2025; Huang et al., 2025). Beyond entry selection, attention statistics guide budget allocation and retention policies across layers and heads (Cai et al., 2024b; Feng et al., 2026; Ge et al., 2024; Wan et al., 2025), and can also support selection and merging in multimodal models (Wan et al., 2024; Li et al., 2025). KVzip (Kim et al., 2026a) scores entries using attention during context reconstruction, while StreamMem (Yang et al., 2025b) uses attention from generic auxiliary queries to compress streaming video memory.

Other compression strategies. Other approaches exploit positional or temporal structure, representation redundancy, or reduced precision. StreamingLLM (Xiao et al., 2024) retains initial

Table 1: Top-k KV selection overlap and retention of KV entries at gold-document positions. All overlap and retention values are reported as percentages.

Retention Ratio	KV entries overlap (%)		Gold-document KV retention (%)	
	Proxy 1–Proxy 2	Question–Proxy 1	Question	Proxy 1
0.1	75.2	44.9	16.8	7.4
0.3	81.0	54.0	41.3	25.9
0.5	85.9	73.0	60.9	45.4
0.7	90.0	83.4	78.0	66.1
0.9	96.2	93.8	93.2	88.2

attention-sink tokens and a recent sliding window, while KeyDiff (Park et al., 2026) selects distinctive entries using key similarity. For streaming video, temporal redundancy and value norms guide entry selection under a fixed memory budget (Kim et al., 2026b). Other strategies include merging across adjacent layers (Liu et al., 2024b), low-rank projections (Chang et al., 2025), and quantization (Liu et al., 2024c; Hooper et al., 2024; Kang et al., 2024). Dynamic Memory Compression (Nawrot et al., 2024) learns whether to append new KV entries or merge them with the most recent cached state.

Although attention provides a natural signal for KV eviction, its magnitude does not necessarily directly reflect query-specific importance. Our analysis reveals shared attention patterns across different queries over the same context. This observation suggests that raw attention scores mix query-shared structure with query-specific signals, motivating us to distinguish these components when estimating KV-entry importance.

3 Attention Exhibits Both Query-Invariant and Query-Specific Patterns

Many KV-cache eviction methods use attention scores as a proxy for the importance of cached entries (Li et al., 2024a; Kim et al., 2026a). However, attention sinks (Xiao et al., 2024) illustrate that high attention does not necessarily indicate semantic relevance: initial tokens can absorb attention mass under softmax normalization, and retaining these tokens helps maintain stable streaming inference. Motivated by this phenomenon, we investigate whether query-invariant attention patterns also exist beyond the initial sink positions.

We randomly sample 100 document–question pairs from HotpotQA (Yang et al., 2018) and augment each context with five distractor documents randomly drawn from a separate pool. Holding the context fixed, we compute query-to-context attention with Qwen2.5-1.5B-Instruct (Yang et al., 2024) using the original question and two generic proxy queries: "Summarize the above context." (Proxy 1) and "Translate to French." (Proxy 2). For each query, we select the top-k context KV entries by attention score and examine two aspects of the resulting selections. First, we measure selection overlap between the two proxies and between the original question and Proxy 1 to quantify patterns shared across queries. Second, we compare gold-document KV retention—the fraction of KV entries at gold-document positions that are retained—between the original question and Proxy 1 to assess whether the original question favors entries from the relevant documents.

Different queries share high-attention patterns. As shown in Table 1, the two proxy queries select many of the same KV entries despite requesting different operations: summarization and translation. When retaining only 10% of the context KV entries, their selections overlap by 75.2%, compared with an expected overlap of 10% for independent uniform selections. The original question also shares 44.9% of its selected entries with the Proxy 1 at this retention ratio. These results suggest that attention-based KV selection contains a substantial component shared across the evaluated queries. Figure 1 provides complementary evidence: the original question and both proxies exhibit several attention peaks at similar context KV positions. These shared peaks occur well beyond the initial tokens, indicating that the observed similarities extend beyond the initial sink positions. Consistent with this observation, StreamMem (Yang et al., 2025b) reports overlapping

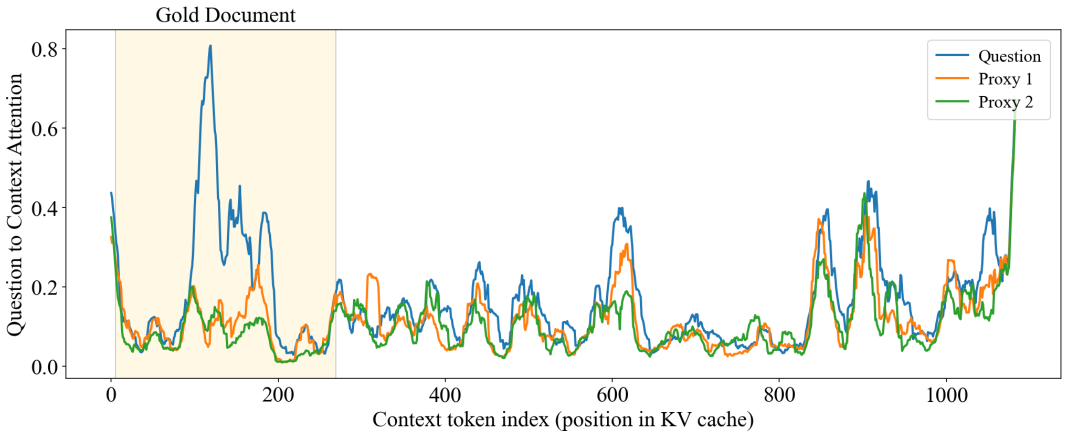


Figure 1: Question-to-context attention for the original question and two unrelated proxy queries. Each curve shows the maximum attention across query tokens, layers, and query heads, smoothed with a 15-token window for visual clarity. The shaded region marks the gold document containing the answer.

attention patterns across different video queries and uses chat-template tokens as a generic query proxy to guide KV-cache compression without access to downstream questions.

Query-specific attention favors relevant documents. Alongside these shared selections, conditioning on the original question consistently increases retention of KV entries from the gold documents. At a retention ratio of 0.1, the original question retains 16.8% of gold-document KV entries, compared with 7.4% for Proxy 1; this advantage persists across all evaluated ratios. Consistent with these results, Figure 1 shows higher attention within the gold document under the original question than under either proxy. While several peaks outside the gold document are prominent across all three queries, attention within the gold document exhibits a pronounced increase when conditioned on the original question. This distinction suggests that comparing question and proxy attention can help identify increases associated with relevant content, even when the attention patterns also contain substantial shared structure. Thus, although different queries prioritize many of the same entries, the original question shifts attention and KV selection toward documents relevant to answering it. Together, these findings suggest that attention contains both query-invariant patterns and query-specific signals associated with relevant information. This motivates contrasting attention under the original and proxy queries to emphasize changes associated with the information needed to answer the question.

4 DeltaKV

4.1 Scoring for KV-Cache Compression

Based on the observations above, we propose **DeltaKV**, a KV-cache compression method that scores each KV entry by contrasting its attention under the real query with that under a proxy query. The key intuition is that attention shared by both queries reflects query-invariant components, whereas their difference highlights attention induced specifically by the real query.

Attention contrast. Let C denote the context, and let q_r and q_p denote the real and proxy queries, respectively. For each KV entry $i = (\ell, h, p)$ at layer ℓ, KV head h, and context position p, we aggregate its attention across query tokens:

$$a_i(q) = \mathrm{Agg}_{t \in q} A_{t,p}^{(\ell,h)}(q, C), \tag{2}$$

where $A_{t,p}^{(\ell,h)}(q, C)$ denotes the attention weight from query token t to context position p, and Agg takes the maximum over query tokens. DeltaKV then computes the attention difference:

$$\Delta a_i = a_i(q_r) - a_i(q_p). \tag{3}$$

Algorithm 1: DELTAKV

Input: Context C, proxy query q_p, real query q_r, model f_{LM}, KV budget B, context KV-entry index set $\mathcal{U}$ indexed by layer ℓ, KV head h, and context position p, sink-entry indices $\mathcal{S} \subseteq \mathcal{U}$, pooling kernel size w

Output: Retained KV-entry indices $\mathcal{K}$

$A(q_p, C) \leftarrow \text{Attn}_{q_p \to C}\big(f_{\text{LM}}([C; q_p])\big)$; // Compute attention using the proxy query

$A(q_r, C) \leftarrow \text{Attn}_{q_r \to C}\big(f_{\text{LM}}([C; q_r])\big)$; // Compute attention using the real query

foreach $q \in \{q_p, q_r\}$ **do**

  foreach $i = (\ell, h, p) \in \mathcal{U}$ **do**

    $a_i(q) \leftarrow \text{Agg}_{t \in q}\, A_{t,p}^{(\ell,h)}(q, C)$; // Aggregate attention over query tokens

  end

  $\bar{a}(q) \leftarrow \text{MeanPool}_{1\text{D}}(a(q), w)$; // Smooth attention scores along context positions

end

foreach $i \in \mathcal{U}$ **do**

  $\Delta\bar{a}_i \leftarrow \bar{a}_i(q_r) - \bar{a}_i(q_p)$; // Compute the difference in smoothed attention

end

$\mathcal{K} \leftarrow \mathcal{S} \cup \text{TopK}_{i \in \mathcal{U} \setminus \mathcal{S}}\,(\Delta\bar{a}_i, B - |\mathcal{S}|)$; // Retain sinks and select additional KV entries

return $\mathcal{K}$

Entries receiving similarly high attention under both queries are downweighted by the subtraction. In contrast, a large positive Δa_i indicates that the real query induces substantially more attention to entry i than the proxy query does, providing a signal of query-specific relevance.

Locality-aware scoring. Before computing the attention difference, we incorporate local context into the attention scores by applying one-dimensional mean pooling with a window size w. Specifically, for each query q, we smooth the aggregated attention scores as

$$\bar{a}(q) = \text{MeanPool}\,1\text{D}(a(q), w) \tag{4}$$

We then compute the query-specific contrast using the locality-aware scores:

$$\Delta\bar{a}_i = \bar{a}_i(q_r) - \bar{a}_i(q_p) \tag{5}$$

This pooling incorporates neighboring KV positions into the importance estimation, yielding a smoother and more robust measure of query-specific relevance while preserving local information (Li et al., 2024a).

Attention sinks. As in previous approaches (Xiao et al., 2024; Kim et al., 2026a; Devoto et al., 2025), we always retain the KV entries corresponding to the first s context tokens as attention sinks. Let $\mathcal{S} \subseteq \mathcal{U}$ denote these protected entries. Given a total KV-entry budget B, we allocate the remaining budget to the highest-scoring unprotected entries:

$$\mathcal{K} = \mathcal{S} \cup \text{TopK}_{i \in \mathcal{U} \setminus \mathcal{S}}\,(\Delta\bar{a}_i, B - |\mathcal{S}|)\,. \tag{6}$$

The overall KV eviction procedure is described in Algorithm 1.

4.2 EFFICIENCY

Computation optimization. DeltaKV prefills the context once and reuses the resulting KV cache to compute attention for both the proxy and real queries. A straightforward implementation, shown in Algorithm 1, processes the two queries in separate forward passes. Instead, we consolidate these evaluations into a single forward pass using a custom attention mask and position IDs, analogous to the tree-attention mechanisms used in speculative decoding (Cai et al., 2024a; Li et al., 2024b). Both queries attend to all context positions and causally to tokens within their respective queries, while attention between the two queries is blocked. Each query is assigned the same position IDs it would receive when processed independently following the context. Together, these masking and positional constraints reproduce the attention dependencies of the two separate forward passes, up to numerical differences. We then extract the attention statistics for each query from the combined

Table 2: Computational complexity of KV-cache importance scoring.

Method	Scoring input	Attention complexity
SnapKV	$q_r \to C$	$O(\|C\|\|q_r\|)$
KVzip	Reconstruction chunks $\to C$	$O(\|C\|^2)$
DeltaKV	$(\mathbf{q_p}, \mathbf{q_r}) \to \mathbf{C}$	$\mathbf{O}(\|\mathbf{C}\|(\|\mathbf{q_p}\| + \|\mathbf{q_r}\|))$

forward pass and apply the same difference-based scoring and cache-selection procedure. After scoring, the proxy-query KV states are discarded, while the real-query states are retained for subsequent decoding.

Complexity analysis. For a fixed model architecture, DeltaKV requires query-to-context attention between the proxy and real queries and the context, resulting in $O(|C|(|q_p| + |q_r|))$ computation, excluding context prefill and query self-attention. As summarized in Table 2, SnapKV requires $O(|C||q_r|)$ computation when using the real query, whereas DeltaKV incurs an additional $O(|C||q_p|)$ cost for the proxy query. In practice, the proxy query can be substantially shorter than the real query. For example, DeltaKV can use the generic instruction "Summarize the above context." as the proxy query, resulting in a small $|q_p|$. Thus, when $|q_p| \ll |q_r|$, the additional computation introduced by the proxy query is small, and DeltaKV has a computational cost close to that of SnapKV.

5 Experiments

5.1 Experimental Setup

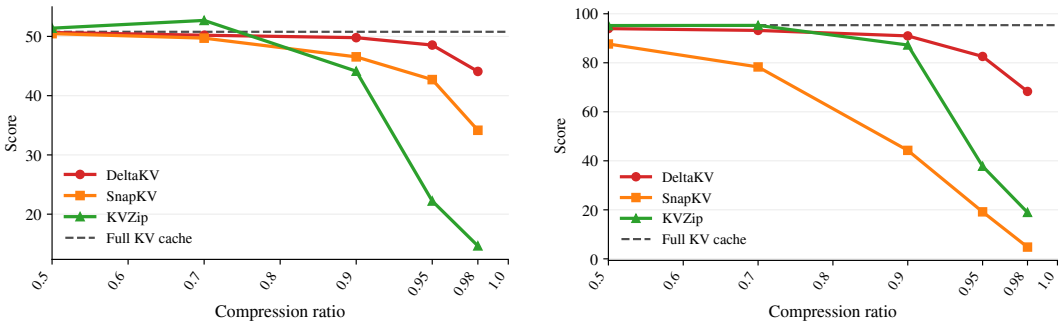
Benchmarks. We evaluate DeltaKV on both text and multimodal long-context tasks. For text, we use LongBench-E (Bai et al., 2024) and RULER (Hsieh et al., 2024). LongBench-E covers real-world tasks such as document question answering, summarization, few-shot learning, and code understanding, while RULER evaluates retrieval, multi-hop tracing, and question answering across varying context lengths. For multimodal evaluation, we use MileBench (Song et al., 2024) and Video-MME (Fu et al., 2025). MileBench tests multi-image reasoning in the presence of distractors, whereas Video-MME evaluates understanding of visual and temporal information in videos of varying durations.

Models. We use Qwen3-8B (Yang et al., 2025a) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) for text tasks, and Qwen3-VL-8B-Instruct (Bai et al., 2025) for multimodal tasks. All methods are evaluated under matched KV-cache budgets.

Baselines. We compare DeltaKV with SnapKV (Li et al., 2024a) and KVzip (Kim et al., 2026a) on text benchmarks, and with SnapKV (Li et al., 2024a), KeyDiff (Park et al., 2026), and StreamingLLM (Xiao et al., 2024) on image and video benchmarks. SnapKV (Li et al., 2024a) scores cached entries using attention from an observation window, which includes the task query in our query-aware setup. KVzip (Kim et al., 2026a) estimates entry importance through context reconstruction. StreamingLLM (Xiao et al., 2024) retains initial attention-sink tokens and a recent sliding window, while KeyDiff (Park et al., 2026) uses key similarity to identify redundancy and retain distinctive entries. We compare methods at matched KV-cache retention ratios and report Full KV performance as an uncompressed reference. Unless otherwise specified, DeltaKV uses "`Summarize the above context.`" as the proxy query and a pooling kernel size of $w = 32$.

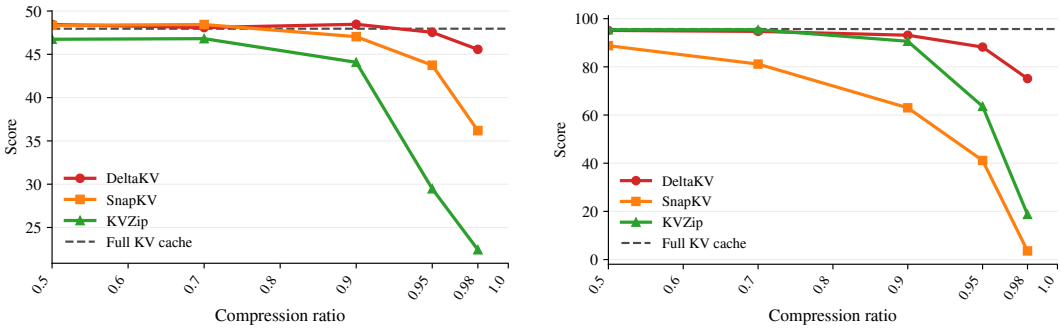
5.2 Results

Figures 2 and 3 summarize the text and multimodal results, respectively, with detailed results for individual tasks and benchmarks provided in Appendix D. Across LongBench-E and RULER, DeltaKV degrades more gracefully under aggressive compression on both Qwen3-8B and Llama-3.1-8B-Instruct. Although SnapKV and KVZip are competitive at moderate compression ratios,



(a) Qwen3-8B on LongBench-E

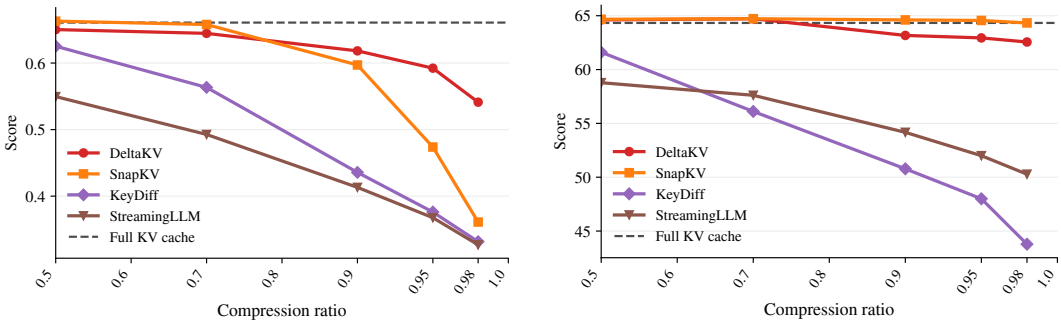
(b) Qwen3-8B on RULER



(c) Llama-3.1-8B-Instruct on LongBench-E

(d) Llama-3.1-8B-Instruct on RULER

Figure 2: Text benchmark performance under varying KV-cache compression ratios. Dashed lines indicate full-KV performance.



(a) MileBench

(b) Video-MME

Figure 3: Multimodal benchmark performance under varying KV-cache compression ratios using Qwen3-VL-8B-Instruct. Dashed lines indicate full-KV performance.

their performance declines more sharply as the KV budget shrinks. DeltaKV consistently outperforms both baselines when compression ratio increases, and its advantage widens as compression increases. For instance, using only 10% of the original KV cache, DeltaKV retains at least 95% of full-cache performance across all evaluated benchmarks and models. These results demonstrate that DeltaKV consistently preserves performance across a wide range of eviction rates, from mild to aggressive, with benefits that generalize across both model families and benchmarks. They also support our hypothesis that discounting attention patterns shared across queries helps prioritize query-relevant information when only a small fraction of the cache can be retained.

This robustness extends to multi-image tasks, where DeltaKV maintains strong performance across a wide range of KV budgets on MileBench, as shown in Figure 3a. Although SnapKV achieves slightly higher scores at moderate compression, its performance drops more sharply under aggressive compression, whereas DeltaKV degrades more gradually. Video-MME exhibits a different

Table 3: Robustness of DeltaKV to proxy query choice on LongBench-E with Qwen3-8B. CR denotes the fraction of KV entries evicted. Percentages in parentheses indicate performance relative to the full-cache baseline (50.78).

Proxy	Input text	CR = 0.90	CR = 0.95
Default	Summarize the above context.	49.79 (98.1%)	48.54 (95.6%)
Proxy 1	The 2026 FIFA World Cup was held in North America, and Spain won the tournament.	49.07 (96.6%)	48.11 (94.7%)
Proxy 2	The Pacific Ocean is the largest ocean on Earth.	50.15 (98.7%)	48.78 (96.1%)
Proxy 3	20 newline characters (\n repeated 20 times)	49.99 (98.4%)	48.81 (96.1%)

Table 4: DeltaKV performance on RULER across model sizes and compression ratios. Full KV denotes decoding without eviction.

	Full KV	Compression ratio (CR)				
Model		0.1	0.3	0.5	0.7	0.9
Qwen3-8B	95.35	95.21	95.17	93.94	93.20	90.97
Qwen3-32B	95.58	95.64	95.60	95.32	95.20	94.79

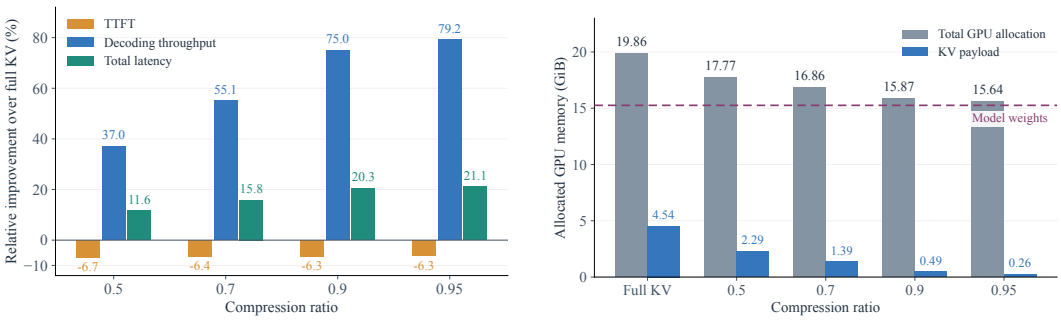
trend in Figure 3b: both SnapKV and DeltaKV remain close to full-cache performance even under aggressive compression, with SnapKV achieving slightly higher scores. One possible explanation is that a 2% KV budget already retains sufficient information to achieve near-full performance in this evaluation setting. Shared attention patterns may also capture useful visual information, potentially contributing to SnapKV's strong performance. This interpretation is consistent with StreamMem (Yang et al., 2025b), which uses chat-template tokens as proxy queries to select visual KV entries based on their attention scores. Its effectiveness provides indirect support for the hypothesis that attention patterns shared across queries can remain useful for KV selection on video inputs.

5.3 Analysis

Robustness to proxy queries. Since DeltaKV relies on a proxy query to identify relevant KV entries, its effectiveness should not depend strongly on a particular choice of proxy. To assess the generalizability of our approach, we evaluate its sensitivity to proxy choice on LongBench-E with Qwen3-8B by replacing the default proxy query with completely unrelated queries. As shown in Table 3, scores vary by only 1.08 and 0.70 points across the four proxies at compression ratios of 0.90 and 0.95, respectively. All variants retain at least 96.6% and 94.7% of full-cache performance at the respective compression ratios. The newline-only proxy retains 98.4% and 96.1% of full-cache performance, suggesting that meaningful natural-language content is not necessary for an effective proxy in this setting. This result provides evidence that the attention patterns used for KV selection may be largely shared across queries, rather than being strongly dependent on the semantic content of the proxy query. Overall, these results support the robustness of DeltaKV to the tested proxy choices.

Model scalability. We evaluate DeltaKV on RULER with 8B- and 32B-parameter models to examine its effectiveness across model sizes (Table 4). Both models remain close to their full-cache baselines under moderate compression, while the 32B model exhibits less degradation as the compression ratio increases. At CR=0.9, the 32B model retains 99.2% of its full-cache score, compared with 95.4% for the 8B model. These results suggest that DeltaKV remains effective across a range of compression ratios, with stronger performance retention observed at the larger model size.

Efficiency. We evaluate DeltaKV's efficiency in terms of decoding throughput, latency, and memory usage. We use Qwen3-8B with BF16 weights and KV cache on a single NVIDIA L40S GPU, using identical synthetic inputs with 32K context tokens, 128 query tokens, and 128 generated tokens, and report medians over 20 runs. Figure 4a shows that higher compression ratios yield larger gains in decoding throughput and greater reductions in total request latency, while TTFT overhead remains nearly constant. At $\mathrm{CR} = 0.90$, DeltaKV improves decoding throughput by 75.0% and



(a) Effect on latency

(b) Effect on GPU memory

Figure 4: Efficiency of DeltaKV with Qwen3-8B and a 32K-token context. (a) Relative improvements in TTFT, decoding throughput, and total latency over full KV; TTFT overhead is shown as negative values. (b) Total allocated GPU memory in the decoding phase and KV-cache payload, with the dashed line indicating memory occupied by model weights.

reduces total request latency by 20.3%, with a 6.3% TTFT increase relative to the full-KV FlashAttention baseline. In addition, Figure 4b illustrates substantial reductions in KV-cache storage, from 4.54 GB to 0.49 GB at $\mathrm{CR} = 0.90$. In general, DeltaKV provides substantial speed and memory benefits.

Compatibility with KV-cache quantization. We evaluate DeltaKV with INT4 KV-cache quantization on LongBench-E. We quantize both keys and values using Quanto [1] with group-wise affine quantization (group size 64, min–max calibration via `MaxOptimizer`), while keeping model weights in BF16. As shown in Figure 5, DeltaKV exhibits similar performance retention with INT4 and BF16 KV caches, preserving over 96% of each precision's no-eviction baseline performance even at 90% eviction. These results show that DeltaKV complements low-bit quantization, providing additional cache reduction through eviction while preserving most of the quantized baseline performance.

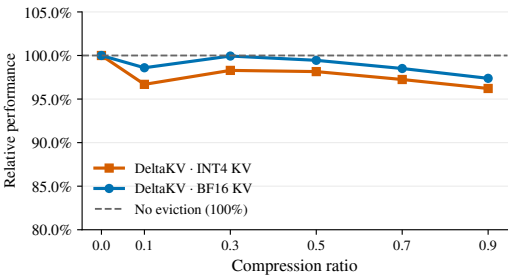


Figure 5: DeltaKV on LongBench-E using Qwen3-8B with BF16 and INT4 KV caches. Performance is relative to each precision's no-eviction baseline.

6 Conclusions

We presented DeltaKV, a KV-cache eviction method motivated by the observation that attention over the same context exhibits substantial shared patterns across different queries. By contrasting real-query attention with proxy-query attention, DeltaKV attenuates shared patterns that can obscure query-specific relevance and emphasizes query-dependent changes. Experiments across four text, image, and video benchmarks demonstrate strong performance retention across a range of eviction ratios. Ablation studies support its robustness across model sizes and design choices as well as its compatibility with KV-cache quantization. Together, these findings highlight query-dependent attention changes as an effective signal for efficient KV-cache compression. These benefits come with limitations: DeltaKV requires the real query before compression, restricting cache reuse across queries or turns, and its current implementation leaves whole-request peak memory unchanged because eviction follows dense context prefill.

[1] https://github.com/huggingface/optimum-quanto

AI USE STATEMENT

We used generative AI tools to assist with identifying relevant literature, implementing code, organizing and presenting experimental results, and drafting and editing the manuscript. The authors reviewed all AI-assisted content and take full responsibility for the code, reported results, and final manuscript.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we describe DeltaKV's attention scoring and KV-selection procedure in Algorithm 1. Section 5.1 specifies the evaluated models, baselines, and benchmarks. Appendix A provides implementation details for KV-cache compression, including method-specific baseline configurations, while Appendix D describes the datasets and evaluation metrics. The hardware configuration and measurement protocol for our efficiency experiments are reported in Section 5.3.

REFERENCES

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. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024a.

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

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed Abdelfattah, and Kai-Chiang Wu. Palu: Kv-cache compression with low-rank projection. In *International Conference on Learning Representations*, volume 2025, pp. 50222–50249, 2025.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025. URL https://arxiv.org/abs/2510.00636.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118. IEEE, 2025.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *International Conference on Learning Representations*, volume 2024, pp. 22975–22988, 2024.

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

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

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

Kai Huang, Hao Zou, Bochen Wang, Ye Xi, Zhen Xie, and Hao Wang. Aircache: Activating inter-modal relevancy kv cache compression for efficient large vision-language model inference. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23958–23967. IEEE, 2025.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm, 2024.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song. KVzip: Query-agnostic KV cache compression with context reconstruction. *Advances in Neural Information Processing Systems*, 38:167563–167591, 2026a.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. Infinipot-v: Memory-constrained kv cache compression for streaming video understanding. *Advances in Neural Information Processing Systems*, 38:138983–139013, 2026b.

Kunxi Li, Yufan Xiong, Zhonghua Jiang, Yiyun Zhou, Zhaode Wang, Chengfei Lv, and Shengyu Zhang. Flowmm: Cross-modal information flow guided kv cache merging for efficient multimodal context inference. *arXiv preprint arXiv:2511.05534*, 2025.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024b.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031, 2024b.

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

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*, 2024c.

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

Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *Advances in Neural Information Processing Systems*, 38:5983–6019, 2026.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Dingjie Song, Shunian Chen, Guiming Hardy Chen, Fei Yu, Xiang Wan, and Benyou Wang. MileBench: Benchmarking MLLMs in long context. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Uhwze2LEwq.

Dezhan Tu, Danylo Vashchilenko, Yuzhe Lu, and Panpan Xu. Vl-cache: Sparsity and modality-aware kv cache compression for vision-language model inference acceleration. In *International Conference on Learning Representations*, volume 2025, pp. 219–239, 2025.

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

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. Look-m: Look-once optimization in kv cache for efficient multimodal long-context inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4065–4078, 2024.

Zhongwei Wan, Hui Shen, Xin Wang, Che Liu, Zheda Mai, and Mi Zhang. Meda: Dynamic kv cache allocation for efficient multimodal long-context inference. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2485–2497, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

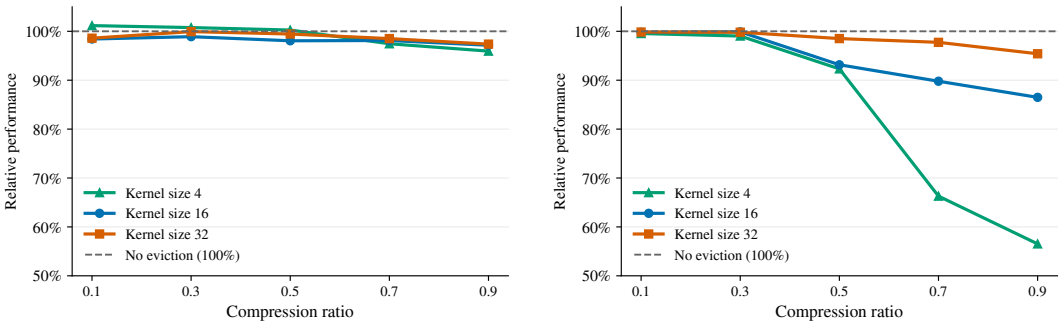
An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. Streammem: Query-agnostic kv cache memory for streaming video understanding. *arXiv preprint arXiv:2508.15717*, 2025b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

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



(a) LongBench-E (b) RULER

Figure 6: Effect of pooling kernel size on DeltaKV with Qwen3-8B on LongBench-E and RULER. Performance is normalized to the corresponding baseline without eviction.

A Implementation Details

We use the KVPress (Devoto et al., 2025) implementations of SnapKV, KVzip, KeyDiff, and StreamingLLM with their default method-specific settings. For SnapKV, we enable query-aware compression by including the actual task query before compression, using the default observation window of 64 tokens and pooling kernel size of 5. KVzip uses reconstruction chunks of 2,048 tokens, applies global pruning across layers, and retains the first four tokens. KeyDiff uses the default key-similarity scoring. StreamingLLM retains the first four tokens as attention sinks and fills the remaining cache budget with the most recent tokens.

We use non-thinking inference for both Qwen3-8B and Qwen3-VL-8B-Instruct. For multimodal evaluations, we use the Qwen3-VL processor to dynamically resize images and video frames while approximately preserving their aspect ratios. For Video-MME, we uniformly sample frames at a target rate of 2 FPS, capped at 768 frames per video.

B Ablation Studies

Effect of pooling kernel size. Figure 6 shows that LongBench-E is relatively insensitive to pooling kernel size, whereas RULER benefits substantially from larger kernels under aggressive compression. On RULER, the degradation with small kernels is concentrated in needle-in-a-haystack subtasks with 32-token needles. This finding highlights task-dependent sensitivity to pooling granularity under aggressive compression. Overall, we use $w = 32$ by default, as it provides greater robustness on RULER while maintaining strong performance on LongBench-E.

C Qualitative Analysis on MileBench

We compare visual KV retention on two MileBench examples at a compression ratio of 0.95, with both methods using the same query for KV selection. In Figure 7, answering the question requires reading the magic number in the middle of the first image. SnapKV predominantly retains recent visual tokens while largely discarding the number-bearing region and produces an incorrect answer, whereas DeltaKV preserves this region and correctly extracts the number. Similarly, in Figure 8, the relevant evidence is the football in the central player's right hand. SnapKV concentrates retention near the bottom of the image while largely discarding the relevant visual evidence and produces an incorrect answer, whereas DeltaKV preserves the region containing the player and the football and answers correctly. These examples are consistent with our observation that attention contains both shared and query-specific patterns, and illustrate how emphasizing query-specific attention helps preserve evidence relevant to the question. The horizontal retention bands in DeltaKV reflect the pooling kernel size: smoothing scores over neighboring positions in the flattened visual-token sequence encourages contiguous retention that appears as horizontal bands in the image.

Question: Deliver concise, accurate answers to questions prompted by the images. What is the special magic Thimphu number in the images?

Original



SnapKV



DeltaKV



Ground truth: 4255900
SnapKV: 4
DeltaKV: The special magic Thimphu number in the images is 4255900.

Figure 7: Qualitative comparison on an ImageNeedle example at $\mathrm{CR} = 0.95$. Rows show the original image and visual KV retention under SnapKV and DeltaKV. Darker regions indicate lower retention.

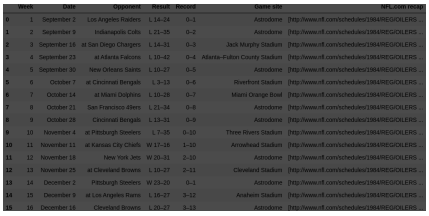
Question: What is the center player holding in his right hand in this photo from the 1984 New Orleans Saints season? Choice list: A. helmet B. water bottle C. football D. towel Your answer is:

Original

	Week	Date	Opponent	Result	Record	Game site	NFL.com recap
0	1	September 2	Los Angeles Raiders	L 14–24	0–1	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
1	2	September 9	Indianapolis Colts	L 21–35	0–2	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
2	3	September 16	at San Diego Chargers	L 14–31	0–3	Jack Murphy Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
3	4	September 23	at Atlanta Falcons	L 10–42	0–4	Atlanta–Fulton County Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
4	5	September 30	New Orleans Saints	L 10–27	0–5	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
5	6	October 7	at Cincinnati Bengals	L 3–13	0–6	Riverfront Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
6	7	October 14	at Miami Dolphins	L 10–28	0–7	Miami Orange Bowl	[http://www.nfl.com/schedules/1984/REG/OILERS ...
7	8	October 21	San Francisco 49ers	L 21–34	0–8	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
8	9	October 28	Cincinnati Bengals	L 13–31	0–9	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
9	10	November 4	at Pittsburgh Steelers	L 7–35	0–10	Three Rivers Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
10	11	November 11	at Kansas City Chiefs	W 17–16	1–10	Arrowhead Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
11	12	November 18	New York Jets	W 20–31	2–10	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
12	13	November 25	at Cleveland Browns	L 10–27	2–11	Cleveland Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
13	14	December 2	Pittsburgh Steelers	W 23–20	0–1	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...
14	15	December 9	at Los Angeles Rams	L 16–27	3–12	Anaheim Stadium	[http://www.nfl.com/schedules/1984/REG/OILERS ...
15	16	December 16	Cleveland Browns	L 20–27	3–13	Astrodome	[http://www.nfl.com/schedules/1984/REG/OILERS ...

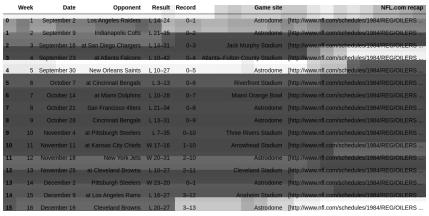


SnapKV





DeltaKV





Ground truth: football
SnapKV: CDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DeltaKV: The center player in the photo from the 1984 New Orleans Saints season is holding a **football** in his right hand. Correct answer: **C. football**

Figure 8: Qualitative comparison on a MultiModalQA example with two input images at $\mathrm{CR} = 0.95$. Rows show the original inputs and visual KV retention under SnapKV and DeltaKV. Darker regions indicate lower retention.

D INDIVIDUAL DATASET PERFORMANCE

We report detailed per-task results for LongBench-E, RULER, MileBench, and Video-MME. For LongBench-E and RULER, we evaluate both Qwen3-8B and Llama-3.1-8B-Instruct. The Qwen3-8B results are presented in Tables 5 and 6, respectively, while the Llama-3.1-8B-Instruct results are presented in Tables 7 and 8. For MileBench and Video-MME, we use Qwen3-VL-8B-Instruct and report results in Tables 9, 10, and 11 for MileBench, and Table 12 for Video-MME. LongBench-E (Bai et al., 2024) evaluates diverse language tasks, using token-level F1 for question answering (including TriviaQA), ROUGE-L for summarization (including SAMSum), accuracy for classification and synthetic tasks, and edit similarity for code completion. RULER (Hsieh et al., 2024) tests retrieval, multi-hop tracing, aggregation, and question answering in controlled long-context settings, using the benchmark's recall-based accuracy. MileBench (Song et al., 2024) assesses temporal and semantic understanding across multiple images, alongside diagnostic retrieval tasks; we report ROUGE-L for open-ended generation and accuracy for diagnostic retrieval. Video-MME (Fu et al., 2025) evaluates video perception, reasoning, and information summarization through multiple-choice accuracy, with our evaluation focusing on medium- and long-duration videos.

Table 5: LongBench-E subtask performance using Qwen3-8B under different KV-cache compression ratios (CR).

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
Qasper	DeltaKV	36.78	36.32	34.96	32.44	28.78
	SnapKV	37.07	36.49	30.51	24.91	18.96
	KVzip	36.06	36.15	29.53	9.64	5.67
MultiFieldQA	DeltaKV	51.72	51.80	49.86	51.09	51.00
	SnapKV	54.62	52.04	47.87	42.83	28.27
	KVzip	55.14	55.08	48.34	25.73	15.29
HotpotQA	DeltaKV	63.82	64.89	64.12	60.82	56.65
	SnapKV	64.14	64.82	62.69	58.60	46.18
	KVzip	64.39	65.17	48.66	11.57	6.74
2WikiMQA	DeltaKV	44.49	44.93	44.28	43.07	37.48
	SnapKV	46.29	44.02	40.15	35.68	29.76
	KVzip	45.48	45.46	38.02	10.65	8.70
GovReport	DeltaKV	31.69	29.54	28.15	25.21	21.60
	SnapKV	32.27	30.45	24.81	21.07	15.10
	KVzip	33.63	33.51	31.05	19.57	11.91
MultiNews	DeltaKV	22.43	21.75	20.55	18.99	16.33
	SnapKV	22.45	21.45	19.12	16.40	12.87
	KVzip	23.55	23.25	22.06	13.01	8.85
TREC	DeltaKV	55.83	59.33	64.67	65.67	58.50
	SnapKV	37.00	33.00	23.67	15.50	6.67
	KVzip	47.33	65.67	55.83	15.00	1.00
TriviaQA	DeltaKV	87.42	86.61	84.91	83.94	78.89
	SnapKV	88.16	88.31	83.29	74.58	53.17
	KVzip	87.73	86.41	87.51	58.46	49.57
SAMSum	DeltaKV	38.40	38.60	37.58	35.40	32.88
	SnapKV	37.79	37.60	36.43	34.47	23.19
	KVzip	38.26	38.07	34.69	19.92	3.96
PassageCount	DeltaKV	23.40	22.83	22.35	20.28	17.00
	SnapKV	19.03	20.00	16.52	15.67	16.74
	KVzip	20.41	20.70	16.00	7.00	0.67
PassageRetrieval	DeltaKV	96.17	95.08	95.61	95.39	80.67
	SnapKV	95.81	96.03	96.05	96.72	89.39
	KVzip	96.61	95.25	56.97	13.44	6.17
LCC	DeltaKV	49.33	44.09	41.93	40.09	32.57
	SnapKV	65.73	65.29	66.11	61.86	50.59
	KVzip	64.88	64.57	45.33	28.13	14.81
RepoBench-P	DeltaKV	57.10	56.79	58.33	58.61	60.81
	SnapKV	55.78	56.74	57.93	56.94	53.16
	KVzip	54.60	55.76	59.74	56.81	57.18

Table 6: RULER subtask performance using Qwen3-8B under different KV-cache compression ratios (CR).

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
NIAH Single 1	DeltaKV	100.00	100.00	100.00	100.00	100.00
	SnapKV	100.00	99.40	57.40	9.80	0.60
	KVzip	100.00	100.00	100.00	100.00	100.00
NIAH Single 2	DeltaKV	100.00	100.00	100.00	100.00	100.00
	SnapKV	100.00	100.00	79.80	1.80	1.20
	KVzip	100.00	100.00	95.60	20.40	0.00
NIAH Single 3	DeltaKV	100.00	99.80	99.60	96.40	44.40
	SnapKV	40.60	8.20	2.40	2.40	2.40
	KVzip	100.00	100.00	98.20	22.80	0.00
NIAH Multikey 1	DeltaKV	100.00	99.80	99.60	99.60	99.40
	SnapKV	100.00	99.80	43.00	10.20	3.00
	KVzip	100.00	99.60	83.60	31.80	0.00
NIAH Multikey 2	DeltaKV	100.00	100.00	99.80	99.20	96.60
	SnapKV	81.60	56.60	15.80	2.00	0.20
	KVzip	100.00	100.00	96.80	45.20	0.00
NIAH Multikey 3	DeltaKV	99.80	99.60	98.80	91.20	26.40
	SnapKV	98.00	78.00	2.00	0.00	0.00
	KVzip	100.00	99.80	93.00	0.00	0.00
NIAH Multivalue	DeltaKV	100.00	99.95	99.70	99.20	89.05
	SnapKV	87.00	59.70	16.00	8.45	2.20
	KVzip	99.95	99.95	79.00	17.35	0.00
NIAH Multiquery	DeltaKV	100.00	100.00	100.00	100.00	99.85
	SnapKV	100.00	100.00	69.45	10.00	2.20
	KVzip	100.00	100.00	97.70	26.80	0.00
Common Words	DeltaKV	95.16	89.28	79.00	13.84	1.46
	SnapKV	98.58	95.94	49.52	24.20	16.00
	KVzip	99.10	99.10	77.88	10.82	0.12
Frequent Words	DeltaKV	93.00	92.20	89.80	86.07	76.53
	SnapKV	93.40	87.80	73.33	61.93	9.53
	KVzip	95.87	96.73	84.27	46.93	0.00
Variable Tracking	DeltaKV	89.20	86.40	73.16	49.48	25.76
	SnapKV	96.88	90.00	37.52	18.44	1.04
	KVzip	100.00	100.00	100.00	100.00	99.68
QA 1	DeltaKV	81.20	82.00	82.00	79.60	73.60
	SnapKV	80.40	80.40	74.20	55.80	2.60
	KVzip	80.00	81.00	74.00	37.60	24.20
QA 2	DeltaKV	62.80	62.60	61.20	59.00	55.00
	SnapKV	63.20	62.00	55.00	44.20	21.40
	KVzip	62.60	62.20	54.20	31.40	22.20

Table 7: LongBench-E subtask performance using Llama-3.1-8B-Instruct under different KV-cache compression ratios (CR).

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
Qasper	DeltaKV	45.15	43.15	43.02	42.60	37.87
	SnapKV	45.48	43.36	34.77	28.01	19.90
	KVzip	46.19	44.73	39.08	16.66	11.70
MultiFieldQA	DeltaKV	55.44	56.37	56.38	55.58	54.93
	SnapKV	55.57	55.42	53.94	45.54	27.90
	KVzip	56.97	56.29	54.88	35.69	19.11
HotpotQA	DeltaKV	60.96	60.57	59.83	58.99	60.53
	SnapKV	60.92	61.35	60.58	57.98	45.24
	KVzip	59.89	57.75	51.73	30.68	23.86
2WikiMQA	DeltaKV	49.74	48.78	50.02	50.52	47.05
	SnapKV	48.81	46.92	46.28	40.63	32.77
	KVzip	49.45	46.39	41.45	23.68	17.90
GovReport	DeltaKV	30.49	29.80	28.50	26.38	23.51
	SnapKV	32.71	30.41	25.14	21.81	15.34
	KVzip	34.92	34.69	31.62	21.90	11.23
MultiNews	DeltaKV	24.38	24.15	23.51	22.09	19.75
	SnapKV	25.16	24.24	21.70	18.64	13.29
	KVzip	25.96	26.10	24.41	18.62	6.88
TREC	DeltaKV	35.00	36.67	42.00	41.33	34.67
	SnapKV	31.67	37.67	31.33	22.00	9.00
	KVzip	10.67	37.33	39.33	10.33	5.00
TriviaQA	DeltaKV	83.09	83.23	83.90	84.57	87.01
	SnapKV	84.17	84.55	82.25	84.17	78.93
	KVzip	84.30	77.40	87.83	88.75	86.17
SAMSum	DeltaKV	38.77	38.56	36.49	34.87	33.30
	SnapKV	37.90	37.52	36.12	34.94	29.06
	KVzip	34.55	33.21	36.67	33.90	30.84
PassageCount	DeltaKV	21.00	19.00	20.67	18.00	14.00
	SnapKV	20.90	20.48	21.33	15.00	9.70
	KVzip	19.11	19.68	16.67	7.00	2.67
PassageRetrieval	DeltaKV	99.67	99.67	99.33	99.33	97.00
	SnapKV	99.67	99.67	99.67	99.67	98.67
	KVzip	99.67	99.67	70.00	21.00	7.00
LCC	DeltaKV	37.67	36.83	35.74	33.27	29.91
	SnapKV	44.44	46.35	53.28	52.80	43.43
	KVzip	43.52	35.78	33.24	25.67	18.19
RepoBench-P	DeltaKV	48.47	48.45	50.75	50.41	52.86
	SnapKV	41.13	41.85	45.09	47.49	47.31
	KVzip	42.42	39.51	46.08	49.24	51.04

Table 8: RULER subtask performance using Llama-3.1-8B-Instruct under different KV-cache compression ratios (CR).

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
NIAH Single 1	DeltaKV	100.00	100.00	100.00	100.00	100.00
	SnapKV	100.00	100.00	100.00	98.80	0.60
	KVzip	99.80	100.00	100.00	100.00	100.00
NIAH Single 2	DeltaKV	100.00	100.00	100.00	99.80	99.80
	SnapKV	100.00	100.00	96.80	88.60	1.20
	KVzip	100.00	100.00	99.60	98.80	1.20
NIAH Single 3	DeltaKV	99.80	99.80	99.60	99.80	92.20
	SnapKV	36.00	8.20	2.40	2.40	2.40
	KVzip	100.00	100.00	100.00	90.60	0.00
NIAH Multikey 1	DeltaKV	99.80	99.80	99.60	99.20	99.20
	SnapKV	99.80	99.80	99.40	89.40	3.20
	KVzip	100.00	100.00	99.20	81.40	0.40
NIAH Multikey 2	DeltaKV	100.00	100.00	99.00	98.60	98.20
	SnapKV	99.20	96.20	84.00	49.20	1.00
	KVzip	100.00	100.00	98.20	72.60	6.00
NIAH Multikey 3	DeltaKV	99.60	99.60	98.80	92.20	41.20
	SnapKV	84.80	40.60	0.20	0.00	0.00
	KVzip	99.80	100.00	96.00	22.80	0.00
NIAH Multivalue	DeltaKV	99.90	99.90	99.70	99.70	94.75
	SnapKV	99.30	98.20	51.25	11.00	1.50
	KVzip	99.95	99.95	93.40	45.65	0.00
NIAH Multiquery	DeltaKV	100.00	99.95	99.80	99.75	99.80
	SnapKV	99.90	99.90	91.45	26.50	1.60
	KVzip	100.00	99.95	98.90	73.50	0.10
Common Words	DeltaKV	95.82	93.26	83.68	42.58	1.48
	SnapKV	97.70	88.32	37.64	4.00	0.02
	KVzip	99.68	99.32	70.20	2.76	0.32
Frequent Words	DeltaKV	94.93	94.33	93.60	90.13	79.00
	SnapKV	90.13	81.53	61.00	32.20	16.27
	KVzip	94.20	94.40	90.33	61.80	0.00
Variable Tracking	DeltaKV	96.32	95.12	89.52	79.92	32.36
	SnapKV	97.56	92.16	53.28	17.88	1.16
	KVzip	99.92	99.92	99.28	97.60	91.04
QA 1	DeltaKV	87.80	88.00	86.00	84.40	78.80
	SnapKV	87.80	88.20	82.40	62.60	0.80
	KVzip	85.60	85.00	78.40	43.80	20.80
QA 2	DeltaKV	62.40	61.60	61.40	60.60	59.80
	SnapKV	62.00	61.80	59.20	52.20	16.80
	KVzip	61.40	62.20	54.60	35.40	23.60

Table 9: MileBench subtask performance using Qwen3-VL-8B-Instruct under different KV-cache compression ratios (CR), part 1 of 3.

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
ALFRED	DeltaKV	0.4178	0.4085	0.2687	0.0693	0.0337
	SnapKV	0.4368	0.3630	0.0627	0.0416	0.0356
	KeyDiff	0.2255	0.1259	0.0805	0.0738	0.0555
	StreamingLLM	0.3836	0.2807	0.1571	0.0883	0.0437
ActionLocalization	DeltaKV	0.4100	0.4050	0.4000	0.3800	0.3750
	SnapKV	0.4300	0.4250	0.4200	0.4200	0.4050
	KeyDiff	0.3950	0.3200	0.3500	0.3200	0.3250
	StreamingLLM	0.4100	0.3700	0.3300	0.3500	0.3200
ActionPrediction	DeltaKV	0.8050	0.7950	0.7600	0.7400	0.7100
	SnapKV	0.8050	0.8050	0.8050	0.8000	0.7200
	KeyDiff	0.7950	0.6900	0.4700	0.3700	0.3250
	StreamingLLM	0.7500	0.6650	0.5750	0.5200	0.4400
ActionSequence	DeltaKV	0.8050	0.8050	0.7850	0.7850	0.7400
	SnapKV	0.7900	0.7950	0.7950	0.8050	0.7050
	KeyDiff	0.7600	0.6750	0.4850	0.3900	0.3150
	StreamingLLM	0.6900	0.6400	0.4750	0.3600	0.2750
CLEVR-Change	DeltaKV	0.2335	0.2335	0.1992	0.0944	0.0647
	SnapKV	0.2461	0.2429	0.0746	0.0868	0.0328
	KeyDiff	0.2204	0.1926	0.1022	0.0962	0.0955
	StreamingLLM	0.2333	0.2385	0.1959	0.1681	0.1350
CharacterOrder	DeltaKV	0.7800	0.7850	0.7750	0.7750	0.7650
	SnapKV	0.8000	0.8050	0.7850	0.7900	0.7750
	KeyDiff	0.7800	0.7650	0.6850	0.6900	0.6400
	StreamingLLM	0.7350	0.7100	0.6600	0.5900	0.5250
CounterfactualInference	DeltaKV	0.5800	0.5250	0.4100	0.5150	0.4350
	SnapKV	0.5950	0.5950	0.5600	0.2300	0.0250
	KeyDiff	0.5700	0.4350	0.3800	0.3650	0.3500
	StreamingLLM	0.3600	0.3350	0.3350	0.3700	0.3900
DocVQA	DeltaKV	0.9500	0.9500	0.9550	0.9550	0.9500
	SnapKV	0.9550	0.9600	0.9600	0.9600	0.9550
	KeyDiff	0.9350	0.9100	0.8050	0.6900	0.6150
	StreamingLLM	0.7500	0.6650	0.5850	0.5800	0.5600
EgocentricNavigation	DeltaKV	0.2700	0.2700	0.2900	0.2750	0.2900
	SnapKV	0.3000	0.3000	0.2950	0.2950	0.2700
	KeyDiff	0.2800	0.2900	0.2750	0.2800	0.2900
	StreamingLLM	0.3000	0.2800	0.2850	0.2700	0.2650
GPR1200	DeltaKV	0.7100	0.7100	0.7067	0.7067	0.6983
	SnapKV	0.7100	0.7100	0.7100	0.6983	0.3983
	KeyDiff	0.7000	0.6450	0.2617	0.1417	0.1200
	StreamingLLM	0.5733	0.5133	0.4367	0.4050	0.3933

Table 10: MileBench subtask performance using Qwen3-VL-8B-Instruct under different KV-cache compression ratios (CR), part 2 of 3.

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
IEdit	DeltaKV	0.1320	0.1222	0.1014	0.0475	0.0243
	SnapKV	0.1285	0.1546	0.0287	0.0282	0.0221
	KeyDiff	0.1028	0.0873	0.0650	0.0664	0.0494
	StreamingLLM	0.1632	0.0992	0.0783	0.0685	0.0585
ImageNeedleInAHaystack	DeltaKV	1.0000	1.0000	1.0000	1.0000	1.0000
	SnapKV	1.0000	1.0000	0.9250	0.7688	0.4219
	KeyDiff	1.0000	1.0000	0.9656	0.7875	0.3344
	StreamingLLM	0.4875	0.2969	0.0719	0.0094	0.0063
MMCoQA	DeltaKV	0.7150	0.7250	0.6850	0.5900	0.4800
	SnapKV	0.6900	0.6650	0.5000	0.3150	0.1200
	KeyDiff	0.5400	0.3200	0.1550	0.1050	0.0950
	StreamingLLM	0.6850	0.6250	0.4950	0.3450	0.2600
MovingAttribute	DeltaKV	0.7700	0.7600	0.7300	0.7200	0.6600
	SnapKV	0.7750	0.7700	0.7600	0.0200	0.0000
	KeyDiff	0.6900	0.6850	0.6150	0.4450	0.3700
	StreamingLLM	0.7950	0.7200	0.6050	0.5500	0.4450
MovingDirection	DeltaKV	0.5500	0.5350	0.5150	0.4900	0.4750
	SnapKV	0.5450	0.5450	0.5450	0.2750	0.0000
	KeyDiff	0.5250	0.4400	0.3400	0.2500	0.2800
	StreamingLLM	0.4900	0.5150	0.3450	0.2950	0.2700
MultiModalQA	DeltaKV	0.7650	0.7550	0.7100	0.6850	0.5400
	SnapKV	0.7750	0.7800	0.4600	0.1200	0.0450
	KeyDiff	0.7400	0.6900	0.5650	0.4550	0.4800
	StreamingLLM	0.7650	0.6900	0.5900	0.4900	0.4100
OCR-VQA	DeltaKV	0.5550	0.5350	0.5000	0.4400	0.0900
	SnapKV	0.5600	0.5500	0.1700	0.0350	0.0000
	KeyDiff	0.6050	0.4950	0.0800	0.0150	0.0300
	StreamingLLM	0.5350	0.3750	0.1600	0.0550	0.0250
ObjectExistence	DeltaKV	0.6700	0.6700	0.5900	0.5650	0.5600
	SnapKV	0.6950	0.6950	0.7000	0.0450	0.0250
	KeyDiff	0.6400	0.6200	0.4900	0.4500	0.4650
	StreamingLLM	0.6850	0.6450	0.5850	0.5550	0.4900
ObjectInteraction	DeltaKV	0.8100	0.8100	0.7850	0.7600	0.7400
	SnapKV	0.8000	0.8000	0.8000	0.7950	0.7200
	KeyDiff	0.7450	0.6900	0.4500	0.3900	0.3000
	StreamingLLM	0.7450	0.7000	0.6200	0.5100	0.3500
ObjectShuffle	DeltaKV	0.3550	0.3450	0.3600	0.3700	0.3900
	SnapKV	0.3650	0.3750	0.3750	0.3700	0.3650
	KeyDiff	0.4050	0.3800	0.3600	0.3300	0.3100
	StreamingLLM	0.3750	0.3500	0.3800	0.3650	0.3500

Table 11: MileBench subtask performance using Qwen3-VL-8B-Instruct under different KV-cache compression ratios (CR), part 3 of 3.

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
SceneTransition	DeltaKV	0.8300	0.8350	0.8200	0.8350	0.8050
	SnapKV	0.8600	0.8600	0.8500	0.8600	0.8000
	KeyDiff	0.7250	0.6850	0.7950	0.7650	0.6600
	StreamingLLM	0.7350	0.6750	0.6600	0.6550	0.6600
SlideVQA	DeltaKV	0.7600	0.7550	0.7250	0.6600	0.5850
	SnapKV	0.8150	0.8200	0.8200	0.6400	0.0450
	KeyDiff	0.8000	0.7500	0.5050	0.4000	0.3600
	StreamingLLM	0.5550	0.4750	0.3950	0.3550	0.3200
Spot-the-Diff	DeltaKV	0.1379	0.1261	0.0609	0.0621	0.0816
	SnapKV	0.1791	0.0978	0.0555	0.0422	0.0186
	KeyDiff	0.1614	0.1492	0.1086	0.0957	0.1036
	StreamingLLM	0.1610	0.1322	0.1215	0.1109	0.0475
StateChange	DeltaKV	0.6750	0.6550	0.6450	0.6350	0.5600
	SnapKV	0.7000	0.7050	0.7150	0.7100	0.7250
	KeyDiff	0.6850	0.6300	0.5500	0.5100	0.4350
	StreamingLLM	0.6800	0.6400	0.5550	0.4700	0.4100
TQA	DeltaKV	0.7550	0.7800	0.7650	0.7550	0.6850
	SnapKV	0.7750	0.7750	0.7550	0.7050	0.4850
	KeyDiff	0.7450	0.7200	0.6400	0.5850	0.5650
	StreamingLLM	0.7450	0.6950	0.5800	0.5350	0.4750
TextNeedleInAHaystack	DeltaKV	0.9031	0.8875	0.9000	0.8656	0.6031
	SnapKV	0.9969	0.9969	0.9187	0.8500	0.6625
	KeyDiff	0.9500	0.6125	0.0969	0.0813	0.0219
	StreamingLLM	0.0719	0.0375	0.0219	0.0156	0.0000
WebQA	DeltaKV	0.7350	0.7350	0.7250	0.6700	0.6700
	SnapKV	0.7300	0.7300	0.7150	0.2750	0.0550
	KeyDiff	0.6950	0.7200	0.5900	0.5900	0.5600
	StreamingLLM	0.6250	0.5600	0.5750	0.5150	0.5300
WikiVQA	DeltaKV	0.9100	0.9000	0.8800	0.8450	0.8050
	SnapKV	0.8950	0.8850	0.8800	0.8850	0.7650
	KeyDiff	0.8400	0.7800	0.6250	0.5200	0.4650
	StreamingLLM	0.7450	0.6900	0.5400	0.5150	0.4700
nuscenes	DeltaKV	0.8700	0.8750	0.8850	0.8850	0.8800
	SnapKV	0.8750	0.8750	0.8750	0.8750	0.8750
	KeyDiff	0.8800	0.8350	0.7450	0.6500	0.6000
	StreamingLLM	0.7100	0.6700	0.5700	0.5450	0.5550

Table 12: Video-MME subtask performance using Qwen3-VL-8B-Instruct under different KV-cache compression ratios (CR). The number of examples in each subtask is denoted by n.

Subtask	Method	CR=0.50	CR=0.70	CR=0.90	CR=0.95	CR=0.98
Action Reasoning ($n = 238$)	DeltaKV	60.50	59.24	57.98	55.88	57.14
	SnapKV	60.08	60.08	59.66	59.66	59.66
	KeyDiff	54.62	52.52	45.80	43.28	44.96
	StreamingLLM	57.98	57.56	56.72	53.36	51.26
Action Recognition ($n = 182$)	DeltaKV	64.29	64.29	62.09	62.64	62.09
	SnapKV	64.84	64.84	64.84	64.84	64.29
	KeyDiff	58.24	51.10	48.35	42.86	37.91
	StreamingLLM	54.40	55.49	47.25	47.80	45.60
Attribute Perception ($n = 100$)	DeltaKV	70.00	70.00	71.00	70.00	66.00
	SnapKV	71.00	71.00	70.00	71.00	70.00
	KeyDiff	62.00	61.00	59.00	59.00	53.00
	StreamingLLM	66.00	65.00	63.00	55.00	54.00
Counting Problem ($n = 143$)	DeltaKV	43.36	42.66	40.56	40.56	41.96
	SnapKV	43.36	44.06	43.36	42.66	42.66
	KeyDiff	42.66	38.46	29.37	30.07	30.07
	StreamingLLM	36.36	34.97	35.66	34.97	32.17
Information Synopsis ($n = 241$)	DeltaKV	81.74	81.74	80.50	81.33	80.08
	SnapKV	81.33	81.33	81.74	81.33	81.74
	KeyDiff	78.42	70.95	64.73	62.24	58.92
	StreamingLLM	79.25	78.01	76.76	71.78	64.73
OCR Problems ($n = 82$)	DeltaKV	58.54	59.76	60.98	58.54	59.76
	SnapKV	58.54	58.54	58.54	57.32	58.54
	KeyDiff	62.20	51.22	47.56	47.56	40.24
	StreamingLLM	54.88	47.56	41.46	43.90	39.02
Object Reasoning ($n = 374$)	DeltaKV	63.10	63.64	61.23	61.76	60.70
	SnapKV	63.64	63.64	63.64	63.64	63.10
	KeyDiff	62.57	56.15	54.01	48.13	43.85
	StreamingLLM	59.63	59.63	55.35	53.74	54.81
Object Recognition ($n = 186$)	DeltaKV	66.13	67.74	64.52	65.59	65.59
	SnapKV	66.13	66.13	66.13	66.67	66.13
	KeyDiff	65.05	59.14	52.69	52.15	37.63
	StreamingLLM	60.22	56.99	49.46	46.77	47.85
Spatial Perception ($n = 24$)	DeltaKV	58.33	58.33	66.67	58.33	58.33
	SnapKV	58.33	58.33	62.50	62.50	58.33
	KeyDiff	54.17	50.00	50.00	50.00	50.00
	StreamingLLM	50.00	41.67	50.00	41.67	37.50
Spatial Reasoning ($n = 29$)	DeltaKV	79.31	75.86	72.41	75.86	75.86
	SnapKV	75.86	75.86	75.86	75.86	75.86
	KeyDiff	75.86	75.86	58.62	48.28	48.28
	StreamingLLM	65.52	68.97	65.52	55.17	48.28
Temporal Perception ($n = 37$)	DeltaKV	78.38	75.68	78.38	81.08	72.97
	SnapKV	78.38	78.38	78.38	78.38	78.38
	KeyDiff	75.68	67.57	62.16	48.65	45.95
	StreamingLLM	70.27	67.57	56.76	62.16	64.86
Temporal Reasoning ($n = 164$)	DeltaKV	60.98	62.20	59.76	57.93	59.15
	SnapKV	60.98	60.98	60.37	60.37	60.37
	KeyDiff	56.10	51.22	42.07	43.29	39.02
	StreamingLLM	45.73	44.51	42.68	43.29	43.29