FACTORKV: A MARGINAL COVERAGE SELECTOR FOR LOW-BUDGET KV CACHE COMPRESSION IN LVLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large vision-language models (LVLMs) rely on growing key-value (KV) caches for autoregressive generation, making cache compression important for long visual contexts. Existing methods improve how limited capacity is allocated across attention heads and how individual KV pairs are scored. Yet under tight budgets, top-k selection can retain tokens from answer-relevant regions while still producing answers with missing or incorrect words. Static ranking can favor overlapping content, leaving complementary answer details insufficiently represented. We propose FactorKV, a KV cache selector based on utility-weighted marginal coverage of the full compressible history in key space. Given a host compressor's budget, protected positions, and utility scores, FactorKV first constructs a pool of high-scoring candidates. It then greedily selects the candidate that provides the greatest additional coverage and updates the represented content after each selection. Historical positions already well covered contribute less to subsequent gains, allowing candidates that represent complementary details to gain priority. Each retention decision therefore accounts for both the utility of the historical content and the coverage supplied by the current selected set. FactorKV integrates with existing compressors without requiring retraining or increasing the cache budget. Experiments show gains across multiple cache allocators and LVLMs, particularly under tight budgets. FactorKV repairs some omission and substitution errors made by the original methods, while others remain unresolved. Under low-budget, decoding latency remains comparable to the base methods, and total latency remains lower than FullKV despite additional prefill computation.

1 INTRODUCTION

Large Vision-Language Models (LVLMs) have demonstrated strong capabilities in image captioning, visual question answering, document understanding, and complex visual reasoning. (Liu et al., 2024a; Zhu et al., 2025) As inputs expand to high-resolution images, multiple images, and videos, the number of visual tokens that models must process increases substantially. During prefill, these tokens contribute to the construction of the KV cache; during subsequent autoregressive generation, the model repeatedly accesses the cached history. Reducing the memory and computational costs of the KV cache while preserving visual understanding has therefore become a challenge for efficient LVLM inference.

Existing research on KV cache compression primarily addresses two questions: how to allocate a limited cache budget and which tokens to retain under that budget. SparseMM (Wang et al., 2025) observes that attention heads respond differently to visual information and adjusts the cache budget for each head accordingly. For token selection, many methods estimate candidate importance from attention or key-value statistics and retain the highest-scoring tokens. MixKV (Liu et al., 2026b) further accounts for semantic redundancy in multimodal KV caches by combining importance and diversity, with their relative weights adjusted according to the redundancy of each attention head. These methods provide effective ways to preserve critical visual information under a limited budget.

Under a tight KV budget, recovering part of the answer-bearing text may still fall short of a complete answer. Fig. 1(a) illustrates this in document and scene text: a time interval printed across two lines is reduced to its start time, the two-word wine label "Mount Riley" becomes "Mount Mount," and a multi-line title loses "Four" while replacing "Meals" with "Health." In each case, the pre-

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.

diction contains a correct fragment but misses another detail needed for the full answer. In fixed random samples of 200 validation questions per task at a budget of 64, omission, substitution, and mixed/empty predictions account for 30.5%–60.5% of DocVQA questions and 15.5%–30.5% of TextVQA questions across four allocators with MixKV (Fig.1(b)). These counts measure the prevalence of the answer-error forms; the examples show how line breaks and visual gaps can separate the details on which an answer depends.

These examples motivate a set-dependent view of token value. A one-pass Top-K rule ranks candidates by fixed scores without accounting for what the selected tokens already represent. If several high-ranked tokens encode similar parts of the text, they can consume the limited budget while a distinct end time, second brand word, or title word remains uncovered. This risk is especially relevant when answer components span lines or are separated by large gaps. Selection should therefore value a candidate by the important content it adds to the current cache, rather than by its score alone.

We introduce **FactorKV** to make this contribution explicit. FactorKV takes the budget, protected positions, and candidate scores from an existing compression method and forms an eligible pool of high-scoring tokens. It then evaluates candidates by their marginal, importance-weighted coverage of the *entire compressible history* in key space. As tokens are selected, content already represented by the cache contributes less to the next selection, allowing tokens that cover complementary details to gain priority. FactorKV thus replaces one-pass ranking with a set-dependent selection process while retaining the base method's budget allocation. Fig.1(c) illustrates the selection logic with a split time interval; it depicts the mechanism rather than a measured token trace.

We evaluate FactorKV with four cache allocators on five Qwen2-VL image benchmarks at a budget of 64 as shown in Fig.1(d). It improves on the corresponding base allocator in all 20 allocator–benchmark comparisons and exceeds MixKV in 17 of 20. The comparisons with the strong SparseMM allocator remain mixed relative to MixKV. We also observe gains on InternVL3 and LLaVA-NeXT, particularly at tighter budgets. Integrating FactorKV into the full TGV-KV pipeline improves several image-task results on Qwen3-VL and most Video-TT subtasks when the methods are evaluated with a common judge. On text-only LongBench, gains on NarrativeQA and Qasper do not extend to a higher overall average because multi-step integration and structured generation tasks, including Musique and code completion, favor the corresponding base methods. These results support set-dependent coverage as a useful selection principle for visual KV compression and show that its benefit depends on the task's information structure.

Our contributions are as follows:

(1)**A characterization of incomplete answers under low-budget visual KV compression.** Across DocVQA and TextVQA, we show that predictions can recover answer fragments while omitting, substituting, or repeating decisive text. A random-sample analysis measures how often the corresponding answer-edit categories occur across four allocators. (2)**A set-dependent coverage objective and selection strategy.** FactorKV uses importance-weighted marginal coverage of the full compressible history to update each candidate's value after every selection. It plugs into existing allocators using their budgets, protected positions, and scores. (3)**An evaluation across allocators, models, and tasks.** We test FactorKV on five Qwen2-VL image benchmarks, additional vision-language models, and the full TGV-KV pipeline, and identify the text-only tasks on which its gains do and do not transfer.

2 Related Work

2.1 Large Vision-Language Models

Large Vision-Language Models (LVLMs) typically consist of a visual encoder, a vision-language connector, and an autoregressive language model. The visual encoder produces a sequence of visual representations, which the connector maps into the language model's representation space. The resulting visual tokens are combined with text tokens from questions or instructions for cross-modal understanding and generation. LLaVA exemplifies an architecture that connects a visual encoder to a language model through a projection layer, while Qwen-VL, LLaVA-NeXT, and InternVL3 extend visual understanding and input handling along different directions (Liu et al., 2023; Bai et al., 2023; Liu et al., 2024a; Zhu et al., 2025).

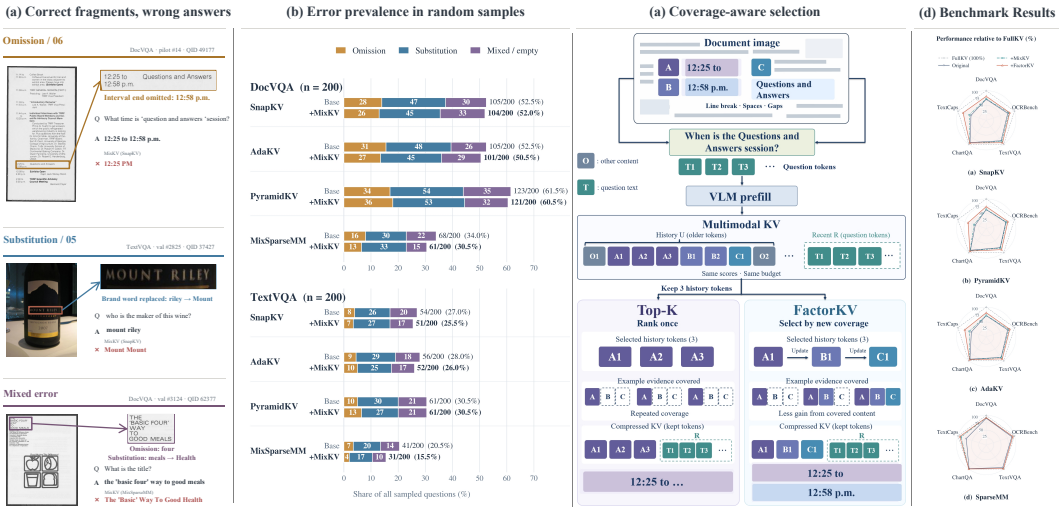


Figure 1: **Motivation and effectiveness of FactorKV** (Qwen2-VL-7B-Instruct, $B = 64$). (a) Illustrative MixKV errors in DocVQA and TextVQA: an omitted interval endpoint, a repeated brand word, and omitted/substituted title words. Boxes denote display crops, not token selections; the interval case is outside the random cohort. (b) Omission-only, substitution-only, and mixed/empty outputs on the same 200 random validation questions per task across four allocators, with and without MixKV. Segment labels are counts; widths are percentages of 200. Exact and insertion-only outputs are omitted. (c) Equal-budget schematic contrasting fixed top-k with FactorKV's marginal coverage of complementary evidence; selections are illustrative. (d) Base, +MixKV, and +FactorKV scores on five full benchmarks. TextCaps uses CIDEr and the other tasks use percentage scores.

As image resolution increases and multi-image, document, and video inputs become more common, the number of visual tokens processed in a single inference run grows. Qwen2-VL produces different numbers of visual tokens for images of different resolutions; Qwen2.5-VL strengthens high-resolution image and long-video processing; and Qwen3-VL further supports interleaved image, text, and video inputs in long contexts (Wang et al., 2024; Bai et al., 2025b;a). During prefill, the model computes attention over the visual and text input and constructs historical Key and Value states at each layer. During autoregressive decoding, newly generated tokens access these cached states without recomputing them. Longer visual sequences therefore increase prefill computation, KV cache memory usage, and cache access during decoding.

2.2 KV Cache Compression

The growth of the KV cache also constrains long-context language models. Compression methods can be understood through two related decisions: how to allocate a retention budget across layers or attention heads, and which historical positions to retain under that budget. H_2O retains positions with high accumulated attention alongside recent positions; StreamingLLM preserves initial attention-sink tokens and a sliding window; and SnapKV uses an observation window near the end of the prompt to estimate which historical KV states will be needed during generation (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024). LookaheadKV uses a learned module to predict future demand for historical positions before eviction, without explicitly generating a proxy answer (Ahn et al., 2026). For budget allocation, PyramidKV adjusts retention across layers, while Ada-KV allocates capacity adaptively across attention heads (Cai et al., 2025; Feng et al., 2025).

Methods for LVLMs additionally account for the different roles of visual and text states. SparseMM identifies vision-related heads from their responses to visual concepts and assigns nonuniform cache budgets across heads (Wang et al., 2025). TGV-KV uses text-visual interactions to determine layer-wise budgets and visual retention priorities while preferentially protecting text KV (Liu et al., 2026a). MixKV combines position importance with Key-based diversity, adjusts their relative weights by attention head, and retains KV states according to the resulting scores (Liu et al., 2026b).

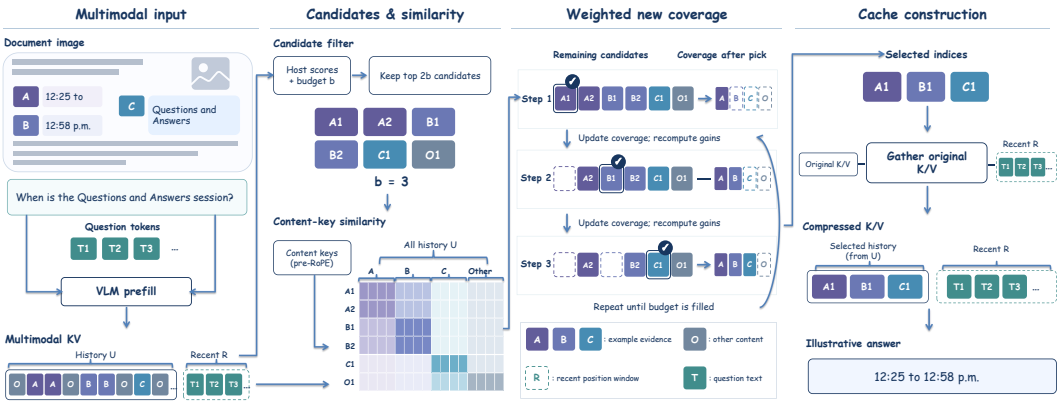


Figure 2: **Overview of FactorKV framework.** Given multimodal input and a cache budget b, FactorKV (1) retains the top $2b$ historical candidates according to the host method's importance scores; (2) computes pre-RoPE content-key similarities and greedily selects candidates by their weighted marginal coverage, updating the covered content after each selection; and (3) gathers the original K/V pairs at the selected positions and combines them with the recent window to construct the compressed cache. In the illustrated example, selecting A1 reduces the gain of the redundant A2, allowing B1 and C1 to preserve complementary evidence for the complete time-range answer.

These methods provide more targeted budgets and position scores, but the final retained set is often determined by a single ranking of candidates.

For text-rich images, selection error may yield a nearly correct textual answer. Retaining information from a relevant sign or document field may leave out a decisive word, repeated text, or the characters needed to distinguish similar strings. Representing the relevant region does not necessarily preserve every detail needed for an exact answer. FactorKV addresses the construction of the retained set after the budget and position scores have been determined. After each selection, it recomputes each candidate's marginal coverage gain over important historical content that the current cache does not yet adequately represent, aiming to retain complementary evidence within a fixed budget.

3 METHOD

FactorKV is a KV selector applied after a host compressor has determined the cache budget and the positions that must be retained. A selection unit may be one KV head or a layer whose KV heads share the same retained positions. For each unit, the host framework supplies a retention budget and protection rules, while a scoring module assigns nonnegative utility weights to the remaining historical positions. FactorKV uses these inputs to select positions with complementary coverage. It constructs the compressed cache after the required prefill scores become available and before decoding begins; the invocation may occur after the full prefill or within a layer-wise prefill procedure. Decoding uses the original Key and Value states at the retained positions.

3.1 BUDGET AND SELECTION PROBLEM

Let g denote a selection unit. For per-head selection, $g = (\ell, h)$; when all KV heads in a layer share retained positions, $g = \ell$. Let $\mathcal{R}_g$ denote its protected positions and $\mathcal{U}_g$ its remaining compressible historical positions, with $N_g = |\mathcal{U}_g|$. Given a total retention budget $B_g \geq |\mathcal{R}_g|$, FactorKV selects $b_g = B_g - |\mathcal{R}_g|$ positions from $\mathcal{U}_g$. If $b_g = 0$, only protected positions are retained. If $N_g \leq b_g$, all historical positions are retained. We omit the index g below.

Each position $j \in \mathcal{U}$ has a nonnegative utility weight w_j. The same weights are used to form the candidate pool and to measure the value of representing historical positions. In the MixKV-series image experiments, we use the combined attention, Value-norm, and Key-diversity score of MixKV as the utility weight. This gives MixKV and FactorKV the same scoring input when comparing their selection rules. Other host frameworks supply the utility scores specified in the integration below.

We clip negative weights to zero and use uniform weights if all weights vanish. We call w_j a *utility weight* because it need not measure attention importance alone.

3.2 Utility-Weighted Coverage of Historical Positions

Candidate positions and the historical positions they represent play different roles. We rank positions in $\mathcal{U}$ by w_j and form a candidate set $\mathcal{C}$ of size

$$m = |\mathcal{C}| = \min(N, \lceil \rho b \rceil), \qquad \rho \geq 1. \tag{1}$$

Only positions in $\mathcal{C}$ can enter the cache, whereas coverage is evaluated over the entire $\mathcal{U}$. The primary configuration uses $\rho = 2$. For heads with very small budgets, a fixed candidate margin can be used; results for this variant are reported separately from those for the primary configuration.

Let $\tilde{\mathbf{k}}_i$ denote the Key representation used for content comparison. We use the pre-RoPE Key when the model provides it; otherwise, we use the Key available through the model interface. The coverage of historical position $j \in \mathcal{U}$ by candidate $i \in \mathcal{C}$ is

$$c_{ij} = \exp\left(\frac{\cos(\tilde{\mathbf{k}}_i, \tilde{\mathbf{k}}_j) - 1}{\tau}\right), \qquad \tau > 0. \tag{2}$$

For a selected set $S \subseteq \mathcal{C}$, the coverage of position j is $q_j(S) = \max_{i \in S} c_{ij}$, with $q_j(\varnothing) = 0$. FactorKV optimizes

$$\max_{\substack{S \subseteq \mathcal{C} \\ |S| \leq b}} F(S), \qquad F(S) = \sum_{j \in \mathcal{U}} w_j q_j(S). \tag{3}$$

Thus, $F(S)$ measures how well the selected positions represent the entire compressible history, weighted by utility. The gain from selecting candidate i first is $F(\{i\}) = \sum_{j \in \mathcal{U}} w_j c_{ij}$. Once a position is well represented, it contributes little additional gain. A candidate's value therefore depends both on the historical positions it covers and on those already covered by the selected set.

An existing set-conditioned selection rule subtracts a candidate's maximum Value similarity to selected positions from its score to limit redundancy within the retained set. Minimal-Intervention KV Retention uses this rule. FactorKV instead evaluates each candidate by its incremental coverage of all historical positions. Two candidates with the same maximum similarity to the selected set can therefore have different gains if they cover different high-utility historical positions.

3.3 Marginal Coverage Selection

Starting from $S_0 = \varnothing$, we compute the marginal gain of each unselected candidate at iteration t:

$$\Delta(i \mid S_t) = F(S_t \cup \{i\}) - F(S_t) = \sum_{j \in \mathcal{U}} w_j \max(0, c_{ij} - q_j(S_t)). \tag{4}$$

We select

$$i_t = \arg\max_{i \in \mathcal{C} \setminus S_t} \Delta(i \mid S_t) \tag{5}$$

and update $S_{t+1} = S_t \cup \{i_t\}$ and $q_j(S_{t+1}) = \max\{q_j(S_t), c_{i_t j}\}$. After b iterations, the selectable budget is filled. The implementation maintains a current coverage vector of length N.

The objective has diminishing returns. If $A \subseteq B \subseteq \mathcal{C}$, then $q_j(A) \leq q_j(B)$. Together with $w_j \geq 0$, this gives $\Delta(i \mid A) \geq \Delta(i \mid B)$. Hence, F is monotone submodular on the fixed candidate pool, and greedy selection achieves the classical $1 - 1/e$ approximation bound for this coverage objective (Nemhauser et al., 1978). This bound applies to the surrogate objective defined on $\mathcal{C}$; it does not directly bound downstream task performance.

3.4 Integration with Host Compressors and Models

FactorKV is inserted at the position-selection step of a host compressor. The host determines the selection unit, cache capacity, protected positions, utility scores, and cache layout; FactorKV selects within the remaining capacity using the coverage objective above. It gathers the original decoding

Key and Value states at the selected positions, leaving the host's budget allocation, protection rules, and decoding-stage cache management unchanged. Model adapters expose the language-decoder cache and, when available, pre-RoPE Keys for content comparison; the visual encoder is not modified. Algorithm 1 gives the selection procedure, while Appendix A.2 specifies the host-dependent inputs and implementation details.

3.5 Cache Construction and Computational Cost

After selection, FactorKV gathers the original Key and Value states at the selected historical positions and combines them with the protected positions in the host's cache layout. The comparison Keys are used only for selection and are not stored in place of the original decoding Keys. The primary configuration uses $\rho = 2$ and $\tau = 0.2$, without a positional-distance factor.

For one KV head, computing similarities between m candidates and N historical positions takes $O(mNd)$ operations, where d is the Key dimension. The resulting $m \times N$ similarity matrix occupies $O(mN)$ space. Direct marginal-gain evaluation over b greedy iterations takes $O(bmN)$ operations and maintains a coverage vector of length N. At the compression step, FactorKV and Top-b retain the same number of positions, $|\mathcal{R}| + \min(b, N)$, under the same budget and protection rules.

4 Experiment

4.1 Experimental Setting

Models and tasks. We primarily evaluate FactorKV on Qwen2-VL-7B-Instruct (Wang et al., 2024) using DocVQA, OCRBench, TextVQA, ChartQA, and TextCaps. To test transfer across architectures, we use LLaVA-NeXT-Mistral-7B (Liu et al., 2024a) and InternVL3-8B (Zhu et al., 2025) with SnapKV. We further integrate FactorKV into TGV-KV (Liu et al., 2026a) on Qwen3-VL-4B/8B-Instruct (Bai et al., 2025a), evaluating image tasks and Video-TT. Text-only transfer is evaluated on the 16 English LongBench tasks with Llama-3.2-1B-Instruct (Meta AI, 2024). FactorKV requires no model fine-tuning. Dataset splits, metrics, and model details are given in Appendix A.3.

Compressors and budgets. On Qwen2-VL, we combine FactorKV with SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2025), AdaKV (Feng et al., 2025), and SparseMM (Wang et al., 2025), and compare each combination with the original compressor and its MixKV variant (Liu et al., 2026b). The Qwen2-VL, LLaVA-NeXT, and InternVL3 experiments use nominal absolute cache budgets $B \in \{64, 128, 256\}$, each including a protected recent window of 32 positions. Qwen3-VL instead follows TGV-KV's input-dependent visual-KV retention targets of 10% and 5%; these percentages are not directly comparable with the absolute budgets. Published base-method and MixKV results serve as reference values, while FactorKV results come from our evaluation.

Selection and evaluation protocol. For SnapKV, PyramidKV, and AdaKV, MixKV and FactorKV receive the same utility scores and host-allocated budgets; their comparison changes the final token-selection rule. The original compressors retain their own scoring rules. FactorKV uses a candidate pool of up to twice the selectable budget and a coverage-similarity temperature of 0.2 by default. The SparseMM integration uses its adaptive head budgets with a 32-position candidate margin. Factor-TGV preserves TGV-KV's budgets, text-guided scores, text retention, and decoding policy while replacing visual-KV selection during prefill; Sparse-FactorTGV additionally selects positions per KV head. For Video-TT, we re-evaluate the compared methods with the DeepSeek-Flash judge. Further details, including the sources of published baselines, are provided in Appendix A.3.4.

4.2 Main results

Compatibility with base compressors. Table 1 compares FactorKV across different base compressors. When combined with SnapKV, PyramidKV, or AdaKV, FactorKV outperforms the corresponding MixKV combination on most evaluated tasks. The gains are more pronounced under smaller cache budgets, particularly on DocVQA, TextVQA, and TextCaps. SparseMM is already a strong base compressor, and FactorKV and MixKV each perform better in different settings when paired with it. These results support FactorKV's compatibility with multiple compressors, while showing that its gains depend on the base method and task.

Table 1: FactorKV with four base KV cache compressors on Qwen2-VL-7B-Instruct. Base-method and MixKV results are from Table 1 of MixKV (Liu et al., 2026b). Shaded rows denote FactorKV; Δ is its difference from MixKV. Report percentages except for TextCaps (raw CIDEr); higher is better.

Methods	DocVQA (%)			OCRBench (%)			TextVQA (%)			ChartQA (%)			TextCaps		
	256	128	64	256	128	64	256	128	64	256	128	64	256	128	64
Qwen2-VL-7B-Instruct															
SnapKV	88.0	80.1	66.5	77.3	71.9	62.4	80.3	77.5	69.9	81.3	79.6	75.5	1.360	1.142	0.794
+ MixKV	90.5	82.6	67.9	79.3	75.4	66.0	81.9	80.6	72.5	81.6	81.2	77.6	1.470	1.342	0.878
+ FactorKV	**92.81**	**88.46**	**76.05**	**80.40**	**77.10**	**68.20**	**82.06**	**81.40**	**76.79**	**81.48**	**80.96**	**77.00**	**1.500**	**1.457**	**1.203**
Δ_{MixKV}	+2.31	+5.86	+8.15	+1.10	+1.70	+2.20	+0.16	+0.80	+4.29	-0.12	-0.24	-0.60	+0.030	+0.115	+0.325
PyramidKV	81.7	74.0	59.9	74.5	67.9	56.8	78.3	74.6	65.3	81.1	79.2	73.5	1.115	0.951	0.569
+ MixKV	84.0	76.3	60.8	76.6	72.6	58.4	80.4	77.1	67.0	81.3	80.7	75.5	1.348	1.119	0.633
+ FactorKV	**90.69**	**84.64**	**69.52**	**77.60**	**74.20**	**62.90**	**81.49**	**79.99**	**72.27**	**81.20**	**80.80**	**76.32**	**1.449**	**1.371**	**0.943**
Δ_{MixKV}	+6.69	+8.34	+8.72	+1.00	+1.60	+4.50	+1.09	+2.89	+5.27	-0.10	+0.10	+0.82	+0.101	+0.252	+0.310
AdaKV	87.4	81.2	67.1	77.8	71.0	62.1	79.9	77.0	70.3	80.8	79.6	75.9	1.345	1.146	0.775
+ MixKV	90.3	82.1	67.8	79.3	74.7	65.5	81.8	79.6	71.2	81.5	80.9	77.4	1.448	1.275	0.878
+ FactorKV	**92.79**	**88.45**	**76.30**	**79.50**	**77.30**	**68.20**	**82.04**	**81.15**	**75.52**	**81.60**	**80.72**	**77.80**	**1.489**	**1.415**	**1.117**
Δ_{MixKV}	+2.49	+6.35	+8.50	+0.20	+2.60	+2.70	+0.24	+1.55	+4.32	+0.10	-0.18	+0.40	+0.041	+0.140	+0.239
SparseMM	93.5	91.5	84.9	81.2	79.0	74.3	82.0	81.6	77.3	82.0	81.5	80.1	1.482	1.430	1.038
+ MixKV	93.8	92.7	86.4	82.0	81.0	77.1	82.0	82.0	80.9	81.6	81.8	81.4	1.480	1.459	1.303
+ FactorKV	**93.79**	**92.52**	**86.83**	**82.30**	**79.80**	**77.40**	**82.07**	**82.18**	**80.77**	**81.84**	**81.48**	**81.28**	**1.492**	**1.483**	**1.374**
Δ_{MixKV}	-0.01	-0.18	+0.43	+0.30	-1.20	+0.30	+0.07	+0.18	-0.13	+0.24	-0.32	-0.12	+0.012	+0.024	+0.071

Table 2: **Conditional recovery of answer errors (Qwen2-VL-7B-Instruct, B = 64).** On the same 200 randomly sampled validation questions per task, cells show the percentage of source errors recovered as normalized exact matches by FactorKV, with recovered/source counts in parentheses. Overall includes insertion-only errors. These conditional rates do not subtract new FactorKV errors.

		DocVQA				TextVQA			
Allocator	Variant	Overall	Omission	Substitution	Mixed/empty	Overall	Omission	Substitution	Mixed/empty
SnapKV	Base	**21.6%** (24/111)	**17.9%** (5/28)	**25.5%** (12/47)	**20.0%** (6/30)	**24.6%** (14/57)	**50.0%** (4/8)	**23.1%** (6/26)	**20.0%** (4/20)
	+MixKV	**20.9%** (23/110)	**26.9%** (7/26)	**22.2%** (10/45)	**15.2%** (5/33)	**23.2%** (13/56)	**28.6%** (2/7)	**25.9%** (7/27)	**23.5%** (4/17)
AdaKV	Base	**25.7%** (28/109)	**19.4%** (6/31)	**35.4%** (17/48)	**19.2%** (5/26)	**22.0%** (13/59)	**44.4%** (4/9)	**24.1%** (7/29)	**11.1%** (2/18)
	+MixKV	**24.1%** (26/108)	**18.5%** (5/27)	**35.6%** (16/45)	**10.3%** (3/29)	**15.1%** (8/53)	**40.0%** (4/10)	**12.0%** (3/25)	**5.9%** (1/17)
PyramidKV	Base	**16.0%** (20/125)	**23.5%** (8/34)	**16.7%** (9/54)	**8.6%** (3/35)	**15.9%** (10/63)	**40.0%** (4/10)	**16.7%** (5/30)	**4.8%** (1/21)
	+MixKV	**15.2%** (19/125)	**22.2%** (8/36)	**15.1%** (8/53)	**6.2%** (2/32)	**9.8%** (6/61)	**15.4%** (2/13)	**11.1%** (3/27)	**4.8%** (1/21)
MixSparseMM	Base	**43.1%** (31/72)	**37.5%** (6/16)	**46.7%** (14/30)	**50.0%** (11/22)	**23.4%** (11/47)	**57.1%** (4/7)	**20.0%** (4/20)	**14.3%** (2/14)
	+MixKV	**38.5%** (25/65)	**30.8%** (4/13)	**45.5%** (15/33)	**33.3%** (5/15)	**2.8%** (1/36)	**25.0%** (1/4)	**0.0%** (0/17)	**0.0%** (0/10)

Recovery of Answer Errors. Across the frozen cohorts of 200 DocVQA and 200 TextVQA questions at $B = 64$, FactorKV turns a subset of non-exact answers from both base and MixKV variants into exact matches for every allocator (Table 2). These recoveries span omissions and substitutions, consistent with better preservation of complementary answer details under a tight KV budget. Recovery remains incomplete and varies by allocator.

Efficiency. As shown in Fig. 3, +FactorKV reduces total latency by 34.2–37.2% relative to FullKV across the four host methods. Compared with each corresponding Base, it adds 0.87–1.47 s to total latency, while decode latency changes by at most 2.3% and peak memory remains unchanged at the reported precision. The separately recorded prefill times account for nearly all of the added latency, indicating that FactorKV's measured overhead occurs primarily during prefill.

Results across different vision-language models. Table 7 evaluates FactorKV across different vision-language models. On InternVL3-8B, FactorKV outperforms SnapKV+MixKV on all five tasks at budgets of 128 and 64 positions. On LLaVA-NeXT-Mistral-7B, it improves most task results at a budget of 64. Results are more mixed for both models at the larger budget of 256. Overall, FactorKV's gains across models are concentrated at tighter cache budgets.

Comparison with full-fledged KV compression methods on advanced VLMs. Table 3 evaluates FactorKV within an existing full KV cache compression method on a newer vision-language model. We integrate FactorKV into the complete TGV-KV pipeline, retaining its layer-wise bud-

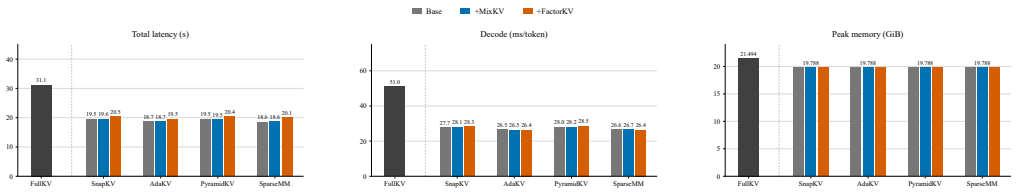


Figure 3: **Efficiency of Qwen2-VL-7B-Instruct at cache budget 64.** FullKV is the uncompressed reference. SnapKV, AdaKV, PyramidKV, and SparseMM are each evaluated in Base, +MixKV, and +FactorKV form. Measurements use one RTX 4090, batch size 1, FP16, FlashAttention 2, a 32,000-token synthetic input without images, and exactly 512 generated tokens. Model loading is excluded.

get allocation, text-guided importance scoring, preferential retention of text KV, and decoding-stage strategy. Only visual KV selection during prefill is replaced. Compared with the full TGV-KV results reported in the original paper (Liu et al., 2026a), both adaptations score higher on DocVQA, TextVQA, and most VizWiz settings, showing that FactorKV's selection also applies to this model and compression framework.

Comprehension on text-only LongBench. We finding gains on comprehension tasks but lower overall scores than the corresponding base methods (see Appendix Table 9).

Video-TT evaluation with a common judge. Table 4 compares the full-cache model, Factor-TGV, and TGV-KV, which is a strong method in the original paper. Because the DeepSeek-V3.2 judging service used in TGV-KV (Liu et al., 2026a) is no longer available, we follow its Video-TT (Zhang et al., 2025) evaluation protocol and re-evaluate all three methods using DeepSeek-Flash with thinking mode disabled. At both 10% and 5% visual KV retention, Factor-TGV outperforms TGV-KV on three of the four subtasks, improving performance on most tasks under this video evaluation setting.

Table 3: Performance of Qwen3-VL at 10% and 5% visual KV retention. Published baselines for ChartQA, DocVQA, VizWiz, TextVQA, and the available 8B TextCaps setting are from TGV-KV paper. We evaluate Factor-TGV and Sparse-FactorTGV on the full evaluation sets. Higher is better; "–" indicates unreported results.

	ChartQA		DocVQA		VizWiz		TextVQA		TextCaps	
Method	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%
					Qwen3-VL-8B-Instruct					
Vanilla	85.3	85.3	95.1	95.1	70.0	70.0	82.1	82.1	33.6	–
StreamingLLM	70.8	72.0	54.6	49.6	63.5	61.3	61.2	54.0	24.5	–
SnapKV	42.3	39.3	80.1	22.6	64.9	53.8	64.0	40.2	27.2	–
H_2O	72.7	67.6	72.0	58.1	62.0	56.6	66.7	58.5	28.8	–
ElasticCache	67.0	67.7	57.0	45.3	53.4	53.6	57.2	48.1	20.1	–
PrefixKV	74.2	71.2	50.3	45.4	57.1	55.7	54.0	49.1	14.4	–
TGV-KV	**82.1**	**73.1**	93.2	88.0	67.6	64.4	79.4	72.0	30.9	–
Factor-TGV	**82.2**	**73.1**	**93.8**	**89.4**	**68.8**	**65.3**	**80.7**	**73.7**	**37.3**	27.9

4.3 Ablation Studies

Selection mechanism. Table 5.I examines which parts of the selection rule matter. FactorKV reports higher scores than one-pass Top-b in all six settings, consistent with the benefit of considering the selected set rather than candidate scores alone. It also exceeds static weighted coverage in all six settings: ranking tokens once by $F(\{i\})$ does not account for coverage already provided by earlier selections. Retaining the set-dependent update but assigning uniform weights produces lower scores in five settings. Finally, FactorKV exceeds both MMR and the α max-similarity penalty in all six settings. These methods adjust a candidate's value according to its similarity to selected tokens, whereas FactorKV evaluates its additional coverage of the entire compressible history. This distinction connects to the incomplete answers in Fig. 1(a): after one text fragment has been represented, a candidate covering a complementary end time or second word

Table 4: Detailed results on Video-TT under 10% and 5% visual KV retention. Vanilla uses the full visual KV cache. Higher is better for all metrics.

	Primary ↑		Correctly-Led ↑		Wrongly-Led ↑		Paraphrase ↑	
Method	10%	5%	10%	5%	10%	5%	10%	5%
Vanilla	21.5		25.8		40.1		23.0	
TGV-KV	19.0	13.4	25.0	**22.8**	35.6	26.0	**21.2**	13.6
Factor-TGV	**20.1**	**14.6**	**25.9**	21.9	**37.0**	**26.1**	20.0	**14.5**

Table 5: Qwen2-VL with SnapKV: selection mechanism, similarity key space, and candidate-pool ratio. Scores are percentages; higher is better. Selection mechanisms are defined in Appendix A.2.1.

Method	**OCRBench**		**TextVQA**		**DocVQA**	
	B64	B128	B64	B128	B64	B128
I. Selection mechanism						
MixKV+SnapKV Top-$b^{\ddagger}$	66.00	75.40	72.50	80.60	67.90	82.60
Static weighted coverage	58.90	66.60	70.66	71.95	73.13	83.70
Uniform-weight coverage	68.00	73.90	75.12	78.62	**76.92**	83.32
MMR	65.50	75.90	72.87	78.88	73.40	84.63
α max-similarity penalty	64.00	71.50	72.33	74.54	75.62	85.59
FactorKV†	**68.20**	**77.10**	**76.79**	**81.40**	76.05	**88.46**
II. Similarity key space ($\rho = 2, \tau = 0.2$)						
Pre-RoPE†	68.20	**77.10**	**76.79**	**81.40**	76.05	**88.46**
Post-RoPE	**68.60**	75.80	75.62	79.42	**78.26**	86.36
III. Candidate-pool ratio						
$\rho = 1.0^{\ddagger}$	66.00	75.40	72.50	80.60	67.90	82.60
$\rho = 1.5$	67.20	76.70	74.35	79.53	**77.66**	86.82
$\rho = 2.0^{\dagger}$	**68.20**	**77.10**	**76.79**	**81.40**	76.05	**88.46**
$\rho = 4$	65.20	75.10	76.20	79.27	75.98	84.86

can retain value. Overall, the reported pattern favors combining candidate weights with updated, full-history marginal coverage.

Similarity key space. FactorKV computes coverage similarity using pre-RoPE Keys, allowing similar content at different positions to count as represented and leaving room for complementary evidence across line breaks or visual gaps. Table 5.II shows that pre-RoPE scores are numerically higher in four of six settings, including all B128 settings; post-RoPE performs better on OCRBench B64 and DocVQA B64. We therefore use pre-RoPE Keys by default, while these exceptions suggest that position-sensitive similarity can help for some tasks and budgets.

Candidate-pool ratio. At $\rho = 1$, selection reduces to the original score-based top-b; larger pools admit candidates with complementary coverage. Table 5.III shows that $\rho = 2$ scores higher than $\rho = 1$ in all six settings, $\rho = 1.5$ in five, and $\rho = 4$ in all six. We use $\rho = 2$ by default because further expansion shows no consistent gain while increasing selection cost.

Qualitative Analysis of Dispersed Evidence Retention. As shown in Fig. 5, the session title is separated from the two-line time field, and the answer hinges on the lower end-time line. MixKV retains tokens near this line in intermediate layers, but its coverage weakens in later layers. FactorKV continues to cover the end time together with the session title and start time. This pattern is consistent with complementary selection preserving the less consistently retained endpoint, allowing FactorKV to recover the complete time range while the baselines substitute an incorrect end time.

5 Conclusion

Under tight KV-cache budgets, existing compressors can retain part of the answer-bearing evidence yet still produce answers with missing or incorrect words. FactorKV addresses this problem by selecting tokens according to their utility-weighted marginal coverage of the full compressible history in key space. This set-dependent selection favors complementary evidence without increasing the cache budget or requiring retraining. Answer-error analysis shows that FactorKV repairs some omission and substitution errors. Results across multiple budget allocators, models, and visual tasks show gains particularly under tight budgets. The additional computation is concentrated in prefill: decode latency remains nearly unchanged, while end-to-end latency increases modestly relative to the corresponding base methods. Future work will improve FactorKV's performance on text-only tasks and examine its use in multi-turn conversations and long-output generation.

AI USE STATEMENT

In this work, we used generative AI tools to a limited extent to polish the writing, identify potentially relevant literature, assist with research ideation or execution, and prepare preliminary drafts of sections of the paper. We have not used generative AI tools for other tasks requiring disclosure, and the remaining required disclosure categories are not applicable to this work. Additionally, we have not used generative AI tools for other tasks with recommended disclosure. We reviewed and revised all AI-assisted drafts, checked identified references against their original sources, and independently evaluated AI-assisted research work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. Lookaheadkv: Fast and accurate kv cache eviction by glimpsing into the future without generation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2603.10899.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. URL https://arxiv.org/abs/2308.12966.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

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 (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 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/.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2406.02069.

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. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2407.11550.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

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. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.14469.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, ocr, and world knowledge, 2024a. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Jizhihui Liu, Ruizi Han, Miao Zhang, Rui Shao, Xuebo Liu, Weili Guan, and Yaowei Wang. Tgv-kv: Text-grounded kv eviction for vision-language models. In *International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2606.03075.

Xuyang Liu, Xiyan Gui, Yuchao Zhang, and Linfeng Zhang. Mixing importance with diversity: Joint optimization for KV cache compression in large vision-language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=B2iqbCQviR.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12), December 2024b. ISSN 1869-1919. doi: 10.1007/s11432-024-4235-6. URL http://dx.doi.org/10.1007/s11432-024-4235-6.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2200–2209, January 2021.

Meta AI. Llama 3.2 Model Card, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. In *European Conference on Computer Vision*, 2020. URL https://arxiv.org/abs/2003.12462.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8309–8318, 2019. doi: 10.1109/CVPR.2019.00851.

Jiahui Wang, Zuyan Liu, Yongming Rao, and Jiwen Lu. SparseMM: Head sparsity emerges from visual concept responses in MLLMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23177–23187, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. URL https://arxiv.org/abs/2409.12191.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17453.

Yuanhan Zhang, Yunice Chew, Yuhao Dong, Aria Leo, Bo Hu, and Ziwei Liu. Towards video thinking test: A holistic benchmark for advanced video reasoning and understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20626–20636, October 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.14048.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025. URL https://arxiv.org/abs/2504.10479.

A APPENDIX

A.1 FACTORKV SELECTION PROCEDURE

Algorithm 1 describes one selection unit, which is either a KV head or a layer with shared retained positions. The host compressor provides the budget, protected positions, utility weights, and original cache states. FactorKV changes the selection of compressible historical positions.

Algorithm 1 FactorKV selection for one host-defined unit

Require: Ordered compressible positions $\mathcal{U} = (u_1, \ldots, u_N)$; protected positions $\mathcal{R}$; total budget $B \geq |\mathcal{R}|$; utility scores $(w_1, \ldots, w_N)$; comparison Keys $(\tilde{\mathbf{k}}_{u_1}, \ldots, \tilde{\mathbf{k}}_{u_N})$; original cache states K, V; candidate ratio $\rho \geq 1$; temperature $\tau > 0$

Ensure: Retained positions $\mathcal{P}$ and compressed cache K', V'

$b \leftarrow B - |\mathcal{R}|$
if $b = 0$ **then**
  return HOSTPACK$(K, V, \mathcal{R})$
end if
if $N \leq b$ **then**
  return HOSTPACK$(K, V, \mathcal{R} \cup \mathcal{U})$
end if
$w_j \leftarrow \max(w_j, 0)$ for every $j \in \{1, \ldots, N\}$
if $\sum_{j=1}^{N} w_j = 0$ **then**
  $w_j \leftarrow 1$ for every $j \in \{1, \ldots, N\}$
end if
$m \leftarrow \min(N, \lceil \rho b \rceil)$
$\mathcal{C} \leftarrow$ TOPM$((w_1, \ldots, w_N), m)$ ▷ Indices into $\mathcal{U}$, not original sequence positions
for each $i \in \mathcal{C}$ and $j \in \{1, \ldots, N\}$ **do**
  $a_{ij} \leftarrow \exp\left(\dfrac{\cos(\tilde{\mathbf{k}}_{u_i}, \tilde{\mathbf{k}}_{u_j}) - 1}{\tau}\right)$
end for
$S \leftarrow \varnothing$; $q_j \leftarrow 0$ for every $j \in \{1, \ldots, N\}$
for $t = 1, \ldots, b$ **do**
  for each $i \in \mathcal{C} \setminus S$ **do**
    $\Delta_i \leftarrow \sum_{j=1}^{N} w_j \max(0, a_{ij} - q_j)$
  end for
  $i^\star \leftarrow \arg\max_{i \in \mathcal{C} \setminus S} \Delta_i$
  $S \leftarrow S \cup \{i^\star\}$
  $q_j \leftarrow \max(q_j, a_{i^\star j})$ for every $j \in \{1, \ldots, N\}$
end for
$\mathcal{P} \leftarrow \mathcal{R} \cup \{u_i : i \in S\}$ ▷ Restore original sequence positions
$(K', V') \leftarrow$ HOSTPACK$(K, V, \mathcal{P})$ ▷ Gather original decoding states in the host's cache layout
return $\mathcal{P}, K', V'$

The candidate set contains high-utility positions, but each candidate's marginal gain is evaluated against all N compressible positions, not just the candidates. The vector q records the best coverage achieved so far; updating it after every choice prevents repeatedly selecting positions that represent the same part of the history. Since $\rho \geq 1$ and this loop is reached only when $N > b$, the candidate set contains at least b positions.

The indices in $\mathcal{C}$ and S refer to the ordered list $\mathcal{U}$. The final mapping $i \mapsto u_i$ restores their original sequence positions. HOSTPACK then gathers the original Key and Value states and applies the host's required ordering and cache layout; comparison Keys are never substituted for stored Keys. For layer-shared selection, a position's comparison feature concatenates the Key features of its KV heads before normalization. A host whose protected positions exceed its budget follows its original fallback rule before invoking Algorithm 1.

Table 6: Host-dependent inputs to FactorKV. The LookaheadKV row describes an insertion interface, not an evaluated integration. The SparseMM-specific variant uses a different utility score and candidate rule from the primary MixKV-series configuration.

Host	Budget and selection unit	Utility and protected positions	FactorKV insertion point
SnapKV (Li et al., 2024)	Fixed historical capacity per KV head.	MixKV combined score in our comparison; recent window protected.	Replaces historical-position Top-K after scoring.
PyramidKV (Cai et al., 2025)	Layer-dependent capacity, applied to each KV head.	Same MixKV combined score; recent window protected.	Replaces selection within each allocated capacity.
AdaKV (Feng et al., 2025)	Adaptive capacity per KV head; variable-length cache retained.	Same MixKV combined score; recent window protected.	Replaces per-head selection after capacity allocation.
SparseMM (Wang et al., 2025)	Visually informed capacity per KV head; variable-length cache retained.	Specialized attention-plus-Value-norm utility; recent window protected.	Replaces per-head selection, using an additive candidate margin.
TGV-KV (Liu et al., 2026a)	Layer-wise capacity; standard Factor-TGV shares selected positions across KV heads.	Text-guided visual scores; text KV receives preferential retention.	Replaces visual-position selection during prefill; the host's insufficient-text-budget fallback and decode policy remain unchanged.
LookaheadKV (Ahn et al., 2026)	Host-defined capacity and protection policy.	Prompt-position scores would come from trained lookahead modules.	Interface specification: replace the subsequent Top-K decision without storing lookahead tokens as prompt KV.

A.2 Host Interfaces and Implementation Details

Table 6 specifies the information supplied by each host and the selection step replaced by FactorKV. The selection procedure itself is shared across hosts; the host remains responsible for allocating capacity and constructing a cache in its required layout.

Budget and degenerate cases. For a selection unit g, the host supplies protected positions $\mathcal{R}_g$ and a total budget B_g. FactorKV receives the selectable budget $b_g = B_g - |\mathcal{R}_g|$ when $B_g \geq |\mathcal{R}_g|$. If the protected positions exceed the budget, the host applies its own fallback before FactorKV is invoked; TGV-KV, for example, retains its original selection fallback in this case. If $b_g = 0$, only protected positions are kept. If the number of compressible positions satisfies $N_g \leq b_g$, all are kept. Negative utility scores are clipped to zero, and all-zero scores are replaced by uniform weights. Because the primary candidate rule uses $\rho \geq 1$, an invocation with $N_g > b_g$ always has at least b_g candidates.

Candidate and similarity configurations. The primary configuration uses $m = \min(N, \lceil \rho b \rceil)$ with $\rho = 2$ and $\tau = 0.2$. The SparseMM-specific variant uses $m = \min(N, b + \delta)$ with $\delta = 32$; its remaining greedy coverage steps are unchanged. For per-head selection, cosine similarity is computed from that KV head's comparison Keys. For the layer-shared TGV-KV selection, the comparison feature at each position concatenates the Keys of all KV heads in the layer before normalization. These features determine selection but are not written into the decoding cache.

Key space, GQA, and index recovery. When an adapter exposes the projected Key before rotary embedding, FactorKV uses it for content comparison; otherwise it uses the Key available through the adapter. In either case, the retained cache contains the model's original decoding Key and Value

states. Under grouped-query attention, attention scores from query heads belonging to one KV head are reduced to a KV-head score before selection; our default reduction is the mean. Candidate and selected indices refer initially to the ordered compressible-position list $\mathcal{U}$. After selection, they are mapped back to original sequence positions, combined with protected positions, and passed to the host's cache-construction routine. This final routine preserves the host-specific ordering and variable-length metadata where applicable.

A.2.1 Selection mechanisms in the SnapKV ablation

Table 5 compares six rules for selecting b tokens within each KV head. Let $w_j \geq 0$ be the fixed host score of token j, and let N be the number of available tokens. For the set-selection rules, the candidate pool is $\mathcal{C} = \mathrm{Top}_m(w)$, where $m = \min\{N, \lceil 2b \rceil\}$. The host scores determine this pool even when a rule changes the weights used inside its objective. For normalized similarity keys k_i, define

$$c_{ij} = \exp\left(\frac{\mathrm{clip}(k_i^\top k_j, -1, 1) - 1}{\tau}\right), \qquad \tau = 0.2, \tag{6}$$

and the weighted coverage of a selected set S as

$$F(S) = \sum_{j=1}^{N} w_j \max_{i \in S} c_{ij}, \tag{7}$$

with zero coverage for $S = \varnothing$. The similarity term uses scoring keys; the chosen indices gather the original host K/V tensors for the cache.

MixKV+SnapKV Top-b. Select the b tokens with the largest host scores w_i. This rule uses neither key similarity nor an iterative set objective and serves as the host-score reference.

Static weighted coverage. Compute each candidate's singleton coverage $F(\{i\})$ once, then select the b candidates with the largest values. Its weights and similarity kernel match those of FactorKV, but selecting a token does not change the scores of the remaining candidates. This comparison tests whether updating residual coverage matters.

Uniform-weight coverage. Apply the same greedy marginal-coverage rule as FactorKV after setting the coverage weights to one: $F_{\mathrm{uni}}(S) = \sum_{j=1}^{N} \max_{i \in S} c_{ij}$. The candidate pool is still formed from the original host scores. This isolates the information carried by w_j in the coverage objective.

MMR. At each step, choose the available candidate maximizing

$$\lambda \widehat{w}_i - (1 - \lambda) \max_{s \in S} c_{is}, \qquad \lambda = 0.7, \tag{8}$$

where $\widehat{w}_i$ is the min–max normalized host score over $\mathcal{C}$ and the maximum is zero for an empty S. MMR updates its redundancy penalty after every choice, without updating coverage over all N tokens.

α max-similarity penalty. First compute the singleton weighted-coverage score $a_i = F(\{i\})$ and min–max normalize it over $\mathcal{C}$ to $\widehat{a}_i$. At each step, choose the available candidate maximizing

$$\alpha \widehat{a}_i - (1 - \alpha) \max_{s \in S} c_{is}, \qquad \alpha = 0.7, \tag{9}$$

again taking the maximum as zero for an empty S. Unlike static coverage, this rule updates a penalty against selected tokens. Unlike FactorKV, its singleton term does not account for which of the N tokens are already covered.

FactorKV. Maintain the current coverage $q_j(S) = \max_{s \in S} c_{sj}$ for every token j. At each step, choose the available candidate with the largest marginal gain

$$\Delta(i \mid S) = F(S \cup \{i\}) - F(S) = \sum_{j=1}^{N} w_j \big[\max\{q_j(S), c_{ij}\} - q_j(S)\big]. \tag{10}$$

After choosing i, update $q_j(S)$ for all tokens. The rule therefore accounts for the coverage supplied by every previously selected token when it scores the next candidate.

A.3 Experimental Details

A.3.1 Models

Qwen2-VL-7B-Instruct. Qwen2-VL-7B-Instruct (Wang et al., 2024) is the primary model for evaluating FactorKV with SnapKV, PyramidKV, AdaKV, and SparseMM as host compressors. Its dynamic-resolution visual processing produces input-dependent numbers of visual tokens, making the retained KV positions consequential for document reading, scene-text recognition, chart understanding, and captioning. We compare each original compressor, its MixKV combination, and its FactorKV combination under the same nominal cache budgets.

LLaVA-NeXT-Mistral-7B. LLaVA-NeXT-Mistral-7B (Liu et al., 2024a) processes images at multiple resolutions through the LLaVA-NeXT visual pathway and uses a Mistral language decoder. We use it to test whether FactorKV transfers beyond Qwen2-VL, with SnapKV as the host compressor.

InternVL3-8B. InternVL3-8B (Zhu et al., 2025) combines a vision encoder, a visual-to-language projection module, and a language decoder, and supports dynamic image processing. As with LLaVA-NeXT-Mistral-7B, its cross-model comparison uses SnapKV as the host compressor.

Qwen3-VL-4B-Instruct and Qwen3-VL-8B-Instruct. The two Qwen3-VL models (Bai et al., 2025a) are used to evaluate FactorKV within the complete TGV-KV pipeline (Liu et al., 2026a). Factor-TGV preserves TGV-KV's layer budgets, text-guided scores, preferential retention of text KV, and decoding-stage strategy while replacing visual KV selection during prefill. Sparse-FactorTGV additionally performs visual-position selection separately for each KV head. Both model sizes are evaluated on image tasks; the Video-TT comparison uses Qwen3-VL-4B-Instruct.

Llama-3.2-1B-Instruct. Llama-3.2-1B-Instruct (Meta AI, 2024) is used for text-only long-context evaluation. We combine FactorKV with SnapKV and with the publicly released trained LookaheadKV modules (Ahn et al., 2026), which provide prompt-position importance scores before cache selection. FactorKV itself requires no model fine-tuning in any experiment.

A.3.2 Image Benchmarks and Metrics

DocVQA. DocVQA (Mathew et al., 2021) asks questions about scanned document images and tests whether a model can locate and read the evidence needed for an answer. We evaluate on the validation split and report Average Normalized Levenshtein Similarity (ANLS), which credits answers according to their normalized string similarity to the references. The local full-set evaluations contain 5,349 questions.

OCRBench. OCRBench (Liu et al., 2024b) evaluates the recognition and use of text appearing in images across a range of OCR-related tasks. We use its 1,000-example evaluation set and report the benchmark's aggregate accuracy.

TextVQA. TextVQA (Singh et al., 2019) requires answers that often depend on reading text in natural images and combining it with the visual context. We use the 5,000-question validation split. Our local evaluator reports case- and punctuation-insensitive exact match, presented as a percentage in the MixKV-series tables.

ChartQA. ChartQA (Masry et al., 2022) tests question answering over charts, including both human-authored and machine-generated questions. We evaluate the 2,500-question test split and report overall relaxed accuracy across its two components.

TextCaps. TextCaps (Sidorov et al., 2020) evaluates captions for images containing text, requiring the generated description to incorporate relevant written content as well as visual information. We use the 3,166-image validation split and report CIDEr. The MixKV-series tables present the raw CIDEr value, whereas the Qwen3-VL table multiplies that value by 100; the two table scales should therefore be read separately.

VizWiz. VizWiz (Gurari et al., 2018) contains photographs and questions submitted by people who are blind, including images that are blurred, poorly framed, or insufficient to answer the question. It is included in the Qwen3-VL image evaluation on the validation split. Our local task configuration reports case- and punctuation-insensitive exact match; baseline values transcribed from TGV-KV retain that paper's "Acc." label.

A.3.3 Video and Long-Context Benchmarks

Video-TT. Video-TT (Zhang et al., 2025) evaluates video understanding and robustness using approximately 1,000 short videos with primary questions and adversarial variants. We report four open-ended conditions separately: the primary question, a correctly led question, a wrongly led question, and a paraphrased question. Responses are scored with an LLM judge. Because the judge used for the published TGV-KV results is unavailable in our setting, we re-evaluate the full-cache model, TGV-KV, and Factor-TGV with DeepSeek-Flash and thinking mode disabled; the Video-TT table contains scores from this common judging protocol.

LongBench. LongBench (Bai et al., 2024) evaluates text-only understanding across long input contexts. We use its 16 English tasks: NarrativeQA, Qasper, and MultiFieldQA-en for single-document question answering; HotpotQA, 2WikiMQA, and Musique for multi-document question answering; GovReport, QMSum, and MultiNews for summarization; TREC, TriviaQA, and SAMSum for few-shot learning; PassageCount and PassageRetrieval-en for synthetic tasks; and LCC and RepoBench-P for code completion. Following the benchmark's task-specific evaluation, question answering uses token-level F1, summarization uses ROUGE, TREC uses classification accuracy, the two synthetic tasks use their counting and retrieval scores, and code completion uses code similarity. The reported average is the arithmetic mean of the 16 task scores. Our FactorKV runs use the English LongBench tasks with FP16 inference and SDPA; the corresponding baseline rows are taken from LookaheadKV's published Llama-3.2-1B-Instruct results.

A.3.4 Evaluation Protocol and Comparisons

Conditional answer-error analysis. The answer-error analysis uses fixed random cohorts of 200 DocVQA and 200 TextVQA validation questions per task at budget 64. We classify non-exact answers by omission, substitution, insertion, and mixed or empty output, and report the fraction of each source method's errors that become normalized exact matches under FactorKV. These conditional recovery rates describe error repair; they are distinct from the full-set benchmark scores.

Definition of answer-error types. We characterize incomplete answers through the difference between a generated answer and its accepted references. We first normalize case, punctuation, and whitespace, then align the prediction with the closest reference using minimum token-level edits. An *omission-only* error ("missing words") omits one or more reference words or numbers without adding or substituting others. A *substitution-only* error replaces reference content with incorrect words or numbers, while an *insertion-only* error adds content without omitting or replacing reference tokens. Predictions requiring more than one edit type, as well as empty answers, belong to the *mixed/empty* category. An answer is recovered when FactorKV produces a normalized exact match to any accepted reference where the compared method did not. These categories describe errors in the generated text; they do not by themselves establish which visual KV positions were lost.

Absolute-budget image experiments. The Qwen2-VL, LLaVA-NeXT, and InternVL3 comparisons follow the MixKV-series setting (Liu et al., 2026b) with nominal cache budgets $B \in \{64, 128, 256\}$. Each budget includes a protected recent window of 32 positions; the host compressor determines how the remaining capacity is allocated across layers or KV heads. For SnapKV, PyramidKV, and AdaKV, the MixKV and FactorKV combinations receive the same combined utility score, so their comparison changes the final position-selection rule. The original, unmodified host methods retain their own scoring rules. Published base-method and MixKV scores are cited from MixKV, while the FactorKV scores are obtained from our evaluation.

Relative-retention image and video experiments. The Qwen3-VL experiments follow TGV-KV's relative visual-KV retention targets of 10% and 5% (Liu et al., 2026a). TGV-KV computes an input-dependent cache target, prioritizes text KV, and allocates the remaining visual capacity across

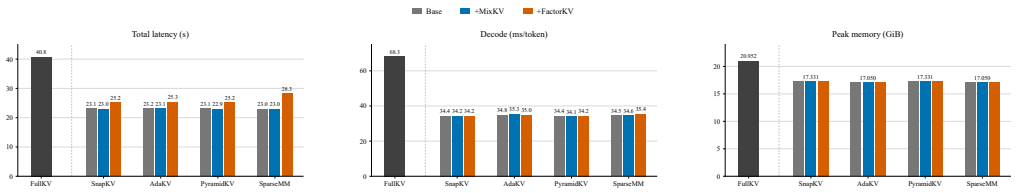


Figure 4: **Efficiency of LLaVA-NeXT-Mistral-7B at cache budget 64.**

Table 7: Cross-backbone comparison under the SnapKV parent compressor. Baseline and +MixKV values follow MixKV Table 1 (Liu et al., 2026b); shaded bold rows report our FactorKV results. Δ denotes FactorKV − MixKV. All metrics are higher-is-better. The first four benchmarks are percentages; TextCaps uses raw CIDEr.

Methods	DocVQA (%)			OCRBench (%)			TextVQA (%)			ChartQA (%)			TextCaps		
	256	128	64	256	128	64	256	128	64	256	128	64	256	128	64
LLaVA-NeXT-Mistral-7B															
SnapKV	59.7	55.2	47.3	45.0	39.0	31.9	63.5	61.0	57.1	50.2	47.5	42.7	0.650	0.558	0.444
+ MixKV	61.7	58.1	48.8	49.9	44.7	36.1	65.2	64.3	60.1	50.8	47.7	43.6	0.708	0.659	0.514
+ FactorKV	**61.03**	**58.25**	**51.71**	**49.70**	**47.10**	**39.60**	**64.81**	**63.28**	**60.59**	**50.56**	**47.96**	**42.72**	**0.676**	**0.657**	**0.554**
Δ_{MixKV}	-0.67	+0.15	+2.91	-0.20	+2.40	+3.50	-0.39	-1.02	+0.49	-0.24	+0.26	-0.88	-0.032	-0.002	+0.040
InternVL3-8B															
SnapKV	89.2	85.4	75.7	80.6	69.0	53.1	80.4	78.2	71.9	86.2	84.6	79.8	1.009	0.901	0.734
+ MixKV	89.4	86.2	76.3	81.9	71.1	52.3	80.9	78.8	72.9	86.3	84.8	80.7	1.029	0.949	0.753
+ FactorKV	**89.95**	**87.39**	**79.76**	**82.60**	**78.70**	**58.90**	**80.70**	**79.45**	**75.47**	**86.00**	**85.88**	**82.80**	**1.012**	**0.988**	**0.850**
Δ_{MixKV}	+0.55	+1.19	+3.46	+0.70	+7.60	+6.60	-0.20	+0.65	+2.57	-0.30	+1.08	+2.10	-0.017	+0.039	+0.097

layers; these percentages therefore describe a different budget convention from the absolute capacities above. The Qwen3-VL image comparison includes the full-cache model, StreamingLLM, SnapKV, H_2O, ElasticCache, PrefixKV, TGV-KV, Factor-TGV, and Sparse-FactorTGV where results are available. Image-task results use the corresponding LMMs-Eval task configurations; results copied from TGV-KV are identified as published baselines in the table captions.

Text-only cache experiments. On Llama-3.2-1B-Instruct, we evaluate SnapKV+FactorKV and LookaheadKV+FactorKV at cache sizes 64, 128, and 256. The LookaheadKV combination keeps the released lookahead modules as its source of importance scores and replaces the subsequent Top-K selection with FactorKV. The published LongBench baselines and our FactorKV results are identified separately in the result tables.

Comprehension gains and reasoning limits on LongBench. To evaluate FactorKV with a trainable compression method, we use Llama-3.2-1B-Instruct, for which the LookaheadKV repository (Ahn et al., 2026) provides trained LookaheadKV weights. Table 9 compares our FactorKV combinations with the baselines reported in Table 9 of LookaheadKV. Both combinations improve on NarrativeQA and Qasper, which emphasize evidence retrieval and text comprehension. They perform worse on multi-hop question answering tasks such as Musique and on code completion, leaving their average scores below the corresponding base methods at all three cache budgets. These results suggest that the current selection strategy better preserves information for direct comprehension than for sequential reasoning. One possible explanation is that semantic coverage, even with utility weights, may treat similar consecutive KV states as redundant although they carry different reasoning steps, thereby omitting necessary intermediate information.

Figure 6 shows the largest co-retention difference in layers 14–20: FactorKV jointly hits the two regions in 15 of 28 slots, versus 8 for SnapKV and 9 for MixKV. This pattern accompanies recovery of the complete category. The advantage is specific to the intermediate band; in layers 21–27, FactorKV records 2 joint hits, compared with 6 and 4 for the two alternatives.

Figure 7 places the answer in a small document header. FactorKV hits this region in 17 and 14 of 28 layer-head slots in the first two bands, compared with four slots in each band for both alternatives.

Table 8: Performance of Qwen3-VL under 10% and 5% visual KV retention. Published baselines for ChartQA, DocVQA, VizWiz, and TextVQA, together with the available 8B TextCaps results, are taken from TGV-KV (Liu et al., 2026a). Factor-TGV and Sparse-FactorTGV are evaluated by us on the full evaluation sets. Higher is better for all metrics.

	ChartQA		DocVQA		VizWiz		TextVQA		TextCaps	
Method	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%
Qwen3-VL-4B-Instruct										
Vanilla	84.1	84.1	93.6	93.6	68.5	68.5	80.6	80.6	–	–
StreamingLLM	67.4	66.9	51.9	47.0	61.5	59.6	56.5	51.7	–	–
SnapKV	23.5	19.8	76.7	12.3	62.1	48.4	58.1	27.0	–	–
H_2O	65.8	63.4	67.7	53.6	62.6	54.8	65.2	55.7	–	–
ElasticCache	68.1	58.8	42.9	33.1	54.0	52.8	47.0	39.8	–	–
PrefixKV	72.9	65.4	52.8	43.0	55.5	49.5	53.0	47.6	–	–
TGV-KV	80.2	**67.2**	91.9	87.3	66.2	63.1	78.1	70.4	–	–
Factor-TGV	80.40	66.68	92.36	**88.79**	66.88	63.42	79.17	72.75	33.48	25.29

Table 9: LongBench results on Llama-3.2-1B-Instruct under different KV-cache budgets. Baseline results are from LookaheadKV (Ahn et al., 2026). Higher is better for all metrics.

	Single-Doc. QA			Multi-Doc. QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
Method	NQA	Qasper	MFQA	Hotpot	2Wiki	Musique	GovRep.	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRetr.	LCC	Repo-P	Avg.
FullKV†	19.24	15.96	42.47	35.53	29.42	19.87	28.34	22.18	25.64	64.00	80.85	38.83	3.00	4.78	38.63	43.35	32.01
KV Cache Size = 64																	
StreamingLLM†	14.42	12.30	24.02	24.04	24.05	9.35	13.56	19.23	13.61	35.50	68.94	29.09	4.00	3.71	37.61	37.33	23.17
SnapKV†	14.24	12.18	30.53	27.30	25.66	12.44	14.27	19.27	12.37	35.00	73.57	29.29	2.00	4.49	38.01	36.84	24.22
PyramidKV†	13.33	11.36	26.05	25.34	24.01	10.57	13.64	19.49	11.96	35.00	69.95	27.75	1.50	4.33	35.25	36.01	22.85
LAQ†	17.21	12.76	37.30	30.27	27.36	14.22	16.42	20.09	14.28	39.50	76.02	31.19	3.00	4.58	39.15	38.37	26.36
LookaheadKV†	17.69	13.30	40.80	33.66	29.80	16.95	18.76	20.65	18.97	45.50	80.12	34.77	2.50	3.22	33.69	36.47	**27.93**
SnapKV + FactorKV‡	16.01	13.50	30.36	25.26	26.85	11.01	15.70	19.66	14.57	35.50	73.05	28.30	3.00	4.00	33.24	32.85	23.93
LookaheadKV + FactorKV‡	**19.03**	**14.26**	38.87	32.26	29.10	13.34	**19.65**	19.41	**18.99**	44.00	79.72	33.83	3.42	4.00	25.87	30.54	26.64
KV Cache Size = 128																	
StreamingLLM†	14.84	12.36	24.67	25.48	23.86	8.73	14.71	19.58	15.50	38.00	71.61	31.82	3.50	3.79	38.81	39.03	24.14
SnapKV†	15.74	12.59	35.87	29.77	26.43	14.17	16.17	20.35	16.47	36.50	78.04	31.84	3.50	4.57	39.03	40.15	26.32
PyramidKV†	14.84	12.10	33.63	27.73	23.95	11.77	15.27	19.79	13.99	35.50	74.17	30.50	1.50	4.71	37.90	37.99	24.71
LAQ†	18.63	13.65	41.78	34.75	29.59	16.57	18.89	20.88	19.19	44.00	79.29	34.89	2.54	4.25	39.43	41.06	28.71
LookaheadKV†	17.38	14.92	41.39	35.46	29.22	17.47	20.13	20.78	21.24	51.50	80.27	36.19	3.00	4.17	34.75	37.68	**29.10**
SnapKV + FactorKV‡	17.20	13.92	34.08	28.38	29.01	11.54	17.96	20.10	18.10	36.50	76.67	32.31	4.00	4.00	33.71	36.56	25.88
LookaheadKV + FactorKV‡	**20.91**	**15.50**	39.81	33.46	**30.76**	15.78	**20.54**	19.65	20.21	50.50	**80.89**	**36.30**	3.02	4.50	26.48	31.56	28.12
KV Cache Size = 256																	
StreamingLLM†	14.74	12.39	25.28	26.71	23.87	8.88	16.99	19.67	17.97	44.50	74.85	35.96	3.50	3.90	40.37	42.19	25.74
SnapKV†	16.59	13.78	38.80	32.54	28.11	16.55	18.55	20.00	19.69	41.50	79.31	33.70	4.00	4.58	39.15	41.26	28.01
PyramidKV†	15.11	13.08	37.31	32.03	25.36	12.60	16.91	20.10	17.78	40.50	76.61	32.34	3.50	4.65	37.13	40.32	26.58
LAQ†	18.31	14.64	41.83	35.34	29.61	17.27	20.68	21.21	21.37	52.00	79.62	36.99	4.04	4.17	40.44	43.37	30.06
LookaheadKV†	18.23	14.70	40.25	36.52	30.45	18.50	21.75	20.91	22.46	57.50	80.09	38.05	4.00	4.50	36.19	40.73	**30.30**
SnapKV + FactorKV‡	19.10	14.39	35.02	30.37	30.16	13.05	19.73	20.51	20.98	45.50	79.62	34.13	3.50	4.50	34.84	36.26	27.60
LookaheadKV + FactorKV‡	**20.66**	**16.48**	40.28	33.22	**31.07**	16.24	21.68	20.24	21.71	55.00	**80.84**	**38.14**	2.79	4.50	28.65	33.73	29.08

† Results reported in Table 9 of (Ahn et al., 2026). ‡ Our evaluation on the 16 English LongBench tasks.

Its total is 77 of 112 slots, versus 49 for SnapKV and 48 for MixKV. The greater retention of the full text region is consistent with the exact transcription.

Figure 8 shows a field-completion error: the digits are correct under every selector, while the suffix differs. FactorKV hits the marked time field in 4 and 7 of 28 slots in the first two bands, compared with 0 and 4 for SnapKV and 1 and 5 for MixKV. Because the box covers both the digits and suffix, these counts describe field-level retention rather than an isolated hit rate for "EST."

In Figure 9, FactorKV's retained tokens extend across both words. The product-name region is hit in 27 and 14 of 28 slots in the first two bands, versus 0 and 0 for SnapKV and 14 and 2 for MixKV. This early coverage accompanies recovery of the complete name.

The challenge in Figure 10 is retaining all three labels in the same layer-head slot. In layers 21–27, FactorKV jointly hits them in 14 of 28 slots, compared with 3 for SnapKV and 5 for MixKV. Across

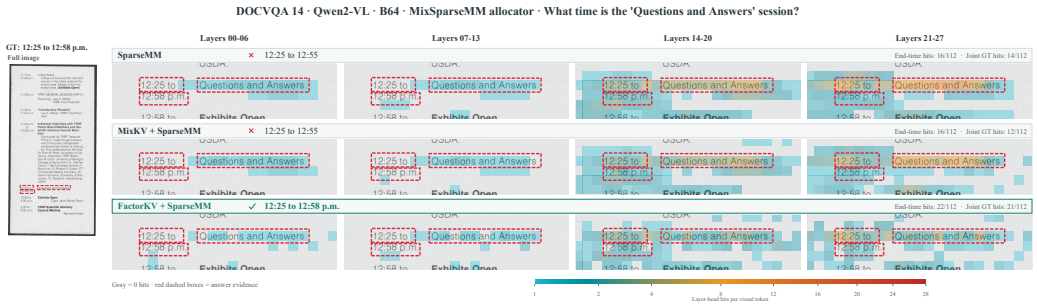


Figure 5: **Cross-line evidence retention in DocVQA #14.** The left panel shows the original document; the three rows show the actual outputs and retained visual tokens of SparseMM, MixKV + SparseMM, and FactorKV + SparseMM for the same Qwen2-VL-7B input, budget of 64, and MixSparseMM layer-head allocation. We map each method's saved KV selection indices back to visual-token positions and, within each seven-layer band, count how many of the 28 layer-head slots retain each token. Color encodes this count, gray indicates no retention, and red dashed boxes mark the session name and the two time endpoints.

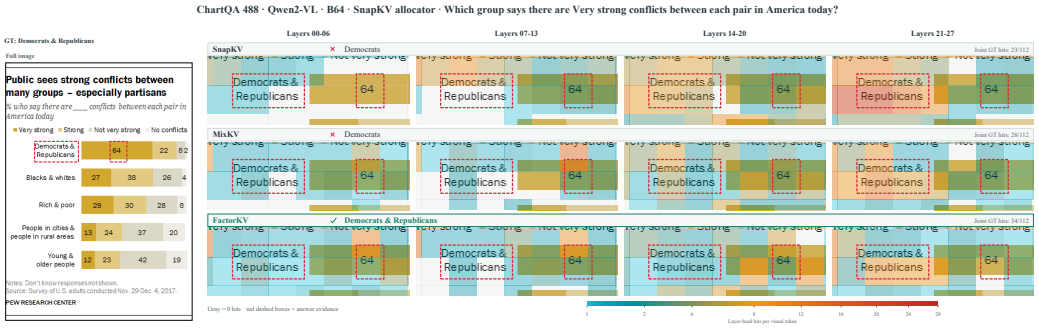


Figure 6: **Complete category and associated value (ChartQA #488).** FactorKV returns "Democrats & Republicans," whereas SnapKV and MixKV return only "Democrats." The marked regions contain the category label and its associated value, 64.

all layers, the joint counts are 32, 6, and 16 of 112 slots, respectively. The difference lies in coverage of the remaining required labels after Germany has been identified.

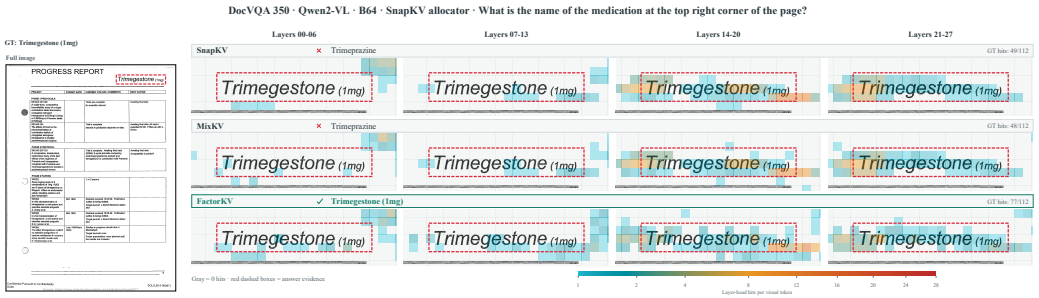


Figure 7: **Small-text medication transcription (DocVQA #350).** SnapKV and MixKV answer "Trimeprazine," whereas FactorKV recovers the annotated "Trimegestone (1mg)."

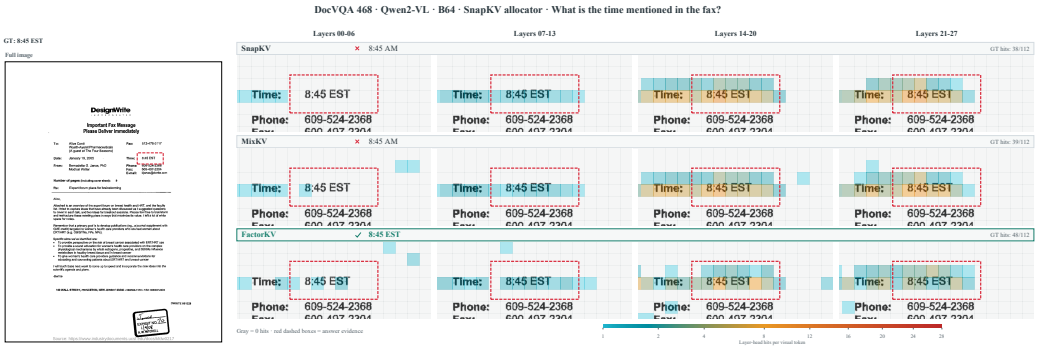


Figure 8: **Completing a fax time field (DocVQA #468).** All selectors recover "8:45," but only FactorKV returns the ground-truth suffix "EST"; SnapKV and MixKV answer "8:45 AM."

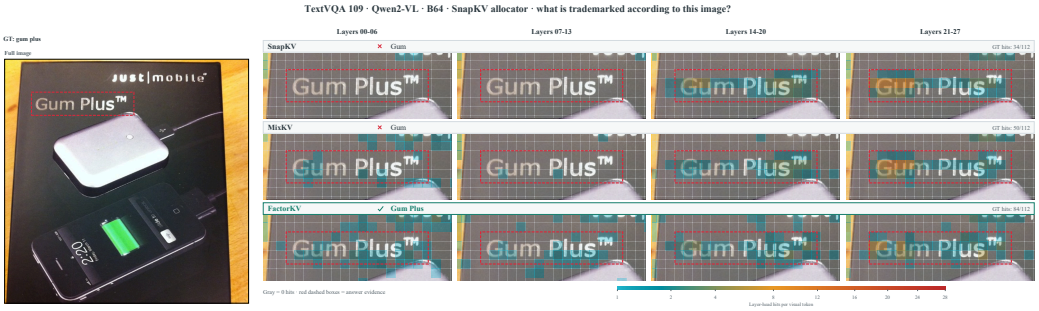


Figure 9: **Two-word product name (TextVQA #109).** FactorKV answers "Gum Plus," matching the ground truth, while SnapKV and MixKV truncate the name to "Gum."

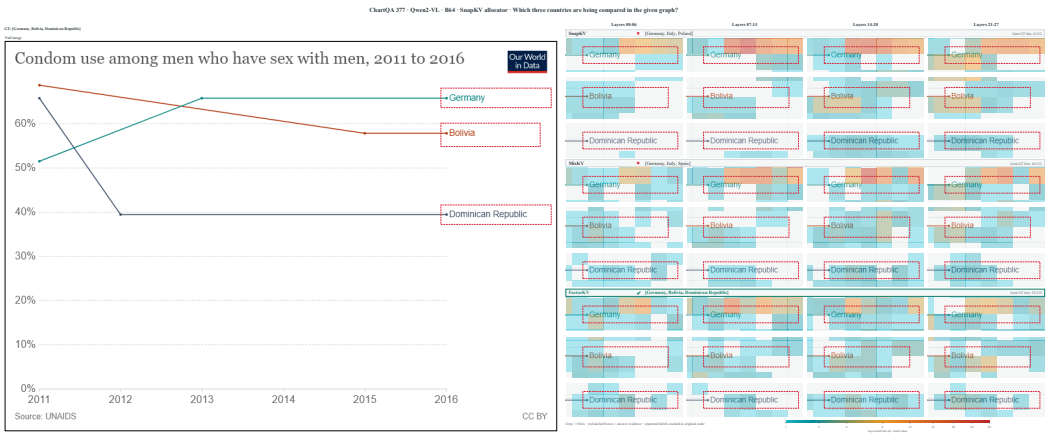


Figure 10: **Three separated chart labels (ChartQA #377).** FactorKV names Germany, Bolivia, and Dominican Republic; SnapKV and MixKV name Germany but replace the other countries. The three evidence strips are displayed in their original vertical order.