HybridKV: Hybrid KV Cache Compression for Efficient Multimodal Large Language Model Inference

Bowen Zeng[1,2]✌, **Feiyang Ren**[1,2]✌, **Jun Zhang**[1,2]✌, **Xiaoling Gu**[3], **Ke Chen**[1,2], **Lidan Shou**[1,2], **Huan Li**[1,2]✉

[1]The State Key Laboratory of Blockchain and Data Security, Zhejiang University
[2]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
[3]Hangzhou Dianzi University, Hangzhou, China

{zbw.cs, feiyangren, zj.cs, chenk, should, lihuan.cs}@zju.edu.cn, guxl@hdu.edu.cn

Abstract

Multimodal Large Language Models (MLLMs) have advanced unified reasoning over text, images, and videos, but their inference is hindered by the rapid growth of key–value (KV) caches. Each visual input expands into thousands of tokens, causing caches to scale linearly with context length and remain resident in GPU memory throughout decoding, which leads to prohibitive memory overhead and latency even on high-end GPUs. A common solution is to compress caches under a fixed allocated budget at different granularities: token-level uniformly discards less important tokens, layer-level varies retention across layers, and head-level redistributes budgets across heads. Yet these approaches stop at allocation and overlook the heterogeneous behaviors of attention heads that require distinct compression strategies. We propose HybridKV, a hybrid KV cache compression framework that integrates complementary strategies in three stages: heads are first classified into static or dynamic types using text-centric attention; then a top-down budget allocation scheme hierarchically assigns KV budgets; finally, static heads are compressed by text-prior pruning and dynamic heads by chunk-wise retrieval. Experiments on 11 multimodal benchmarks with `Qwen2.5-VL-7B` show that HybridKV reduces KV cache memory by up to 7.9× and achieves 1.52× faster decoding, with almost no performance drop or even higher relative to the full-cache MLLM.

1 Introduction

Multimodal Large Language Models (MLLMs) such as `Qwen2.5-VL` (Bai et al., 2025), `LLaVA-OneVision` (Li et al., 2024a), and `InternVL` (Chen et al., 2024c) mark a major step forward in extending large language models beyond text, enabling unified reasoning over both language and visual modalities. To support such capability, they process long multimodal contexts in which each high-resolution image or video frame unfolds into hundreds of tokens, rapidly inflating sequence lengths. This expansion directly drives a linear growth of Key–Value (KV) caches (Zhang et al., 2026), which store the key and value representations of all past tokens for attention computation and must be retained throughout autoregressive decoding. For instance, a 72B-scale `Qwen2.5-VL` processing 20 images already exceeds 40K tokens and 13 GB of cache, while a 5-second 720p video surpasses 50K tokens and 16 GB. Even a small batch of five inputs nearly saturates an entire 80 GB A100 GPU. Such oversized caches impose heavy GPU memory demands and incur substantial latency from repeated memory access during decoding, ultimately limiting the practical use of MLLMs.

To alleviate these memory and latency bottlenecks, several KV cache compression methods

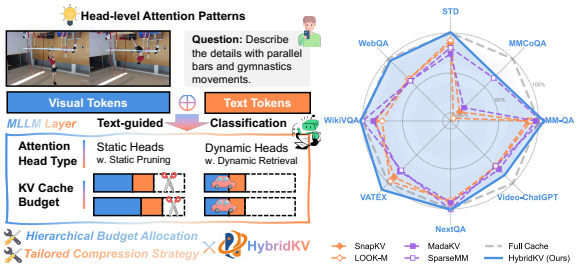


Figure 1: **Left**: We introduce HybridKV, a hybrid KV cache compression framework for efficient yet effective MLLM inference. HybridKV leverages head-level attention patterns (detailed in Section 2) to classify static and dynamic heads via text-guided signals, enabling hierarchical budget allocation with tailored *pruning* and *retrieval* strategies. **Right**: HybridKV outperforms existing counterparts including SnapKV (Li et al., 2024b), LOOK-M (Wan et al., 2024b), MadaKV (Li et al., 2025) and SparseMM (Wang et al., 2025) on eight benchmarks using only 10% of the KV cache on `Qwen2.5-VL-7B`. Performance metrics are shown as a percentage relative to Full Cache.

✌Equal contribution. ✉Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13018–13034

have been proposed for MLLM inference. Under a restricted KV cache budget, these methods aim to optimize budget allocation to achieve generation quality comparable to using the full cache. They can be broadly categorized into token-level, layer-level, and head-level strategies that target different granularities of budget allocation. Token-level methods (e.g., Wan et al. (2024b); Li et al. (2024b); Xiao et al. (2024)) allocate this budget uniformly across layers and heads, discarding unimportant tokens. Layer-level methods (e.g., Li et al. (2025); Wan et al. (2025); Cai et al. (2024)) instead vary the allocation across layers, motivated by the intuition that different layers exhibit different sensitivities to compression amounts, so some layers can be pruned more aggressively while others require more tokens to be preserved. Head-level methods (e.g., Wang et al. (2025); Feng et al. (2025b)) operate at the finest granularity, guided by the intuition that different attention heads capture distinct features or modalities, leading to highly imbalanced importance across heads. While head-level methods push compression to the finest granularity by redistributing budgets across heads, they remain confined to deciding how much each head should be pruned. Yet performance gaps persist, as certain heads tolerate aggressive eviction while others collapse even under mild pruning. This naturally raises a deeper question:

Should accurate KV cache compression stop at budget allocation, or should it also involve ***strategies tailored to different heads****?*

We argue that accurate KV cache compression requires both budget allocation and strategies tailored to different heads. Allocating budgets is necessary but insufficient, because attention heads exhibit distinct behaviors across different stages of inference. We observe that some MLLM heads follow a static pattern, where their KV entries remain stable once established during prefilling and can be pruned safely. Others display a dynamic pattern, where important entries continue to evolve during decoding and cannot be pruned without severe degradation (Section 2.1). This motivates HybridKV, a hybrid compression framework as shown in Fig. 1 (left). HybridKV offers mechanisms to differentiate behaviors of attention heads, and assigns each category a tailored compression method with properly allocated budgets. It operates in three stages. First, since MLLMs interpret visual inputs under textual guidance, we use text tokens as stable anchors and compute a text-centric sparsity score to classify heads into static and dynamic types (Section 3.1). Second, to account for the heterogeneous sensitivity of these heads to compression, HybridKV proposes a top-down budget allocation strategy that hierarchically allocates KV capacity for both different head types and each individual head (Section 3.2). Finally, static heads are compressed with *text-prior pruning*, which retains salient KV entries guided by the attention of local window tokens, while dynamic heads are handled with *chunk-wise retrieval*, which stores their KV cache in blocks and selectively retrieves important chunks during decoding (Section 3.3). Together, these stages enable efficient and accurate multimodal inference, outperforming existing KV cache compression methods on `Qwen2.5-VL-7B` across image and video benchmarks (see Fig. 1 (right)).

To summarize, our main contributions are:

(1) *Novel Pattern Discovery* (Section 2): We perform a systematic analysis of head-level properties within MLLMs, revealing how hybrid attention patterns can guide head-level KV cache compression.

(2) *Hybrid Compression Scheme* (Section 3): We propose HybridKV, an efficient MLLM inference framework that uses hierarchical KV cache allocation and head-specific compression guided by modality-aware features, enabling superior KV cache compression.

(3) *Extensive Experiments* (Section 4): Evaluations on both image and video benchmarks show that HybridKV compresses 90% of KV caches with minimal accuracy loss and delivers up to $1.52\times$ faster decoding on `Qwen2.5-VL-7B`.

2 Observation

2.1 Head-level Attention Patterns

To examine whether distinct patterns exist among attention heads in MLLMs, we conduct a pilot study on `Qwen2.5-VL-7B` using the VATEX (Wang et al., 2019) benchmark[1]. The measure is implemented with a ***cumulative focus count***. For each context token, this metric counts how often it appears among the top-k attended tokens across multiple decoding steps. Formally, for a specific head, the cumulative focus for the j-th token is

$$F_j = \sum_{i=1}^{N} \mathbb{I}(j \in \text{TopK}_{\text{indices}}(\mathbf{a}^{(i)}, k)), \quad (1)$$

where N is the number of decoding steps, $\mathbf{a}^{(i)}$ is the attention score vector over the context at step i,

[1]VATEX provides richly annotated video–text pairs that expand into long multimodal contexts, making it an ideal testbed when combined with a relatively large 7B MLLM.

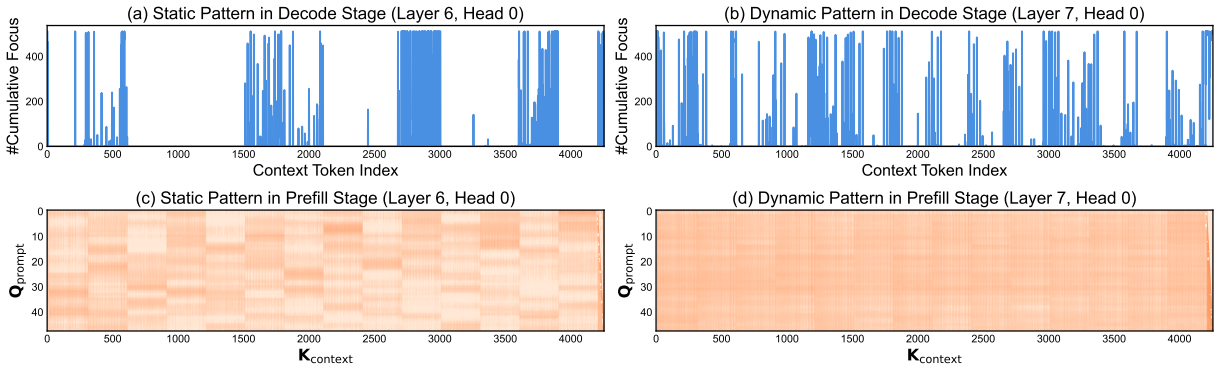


Figure 2: Static vs. dynamic attention patterns in MLLM inference. The decode stage (*a*, *b*) strongly correlates with the corresponding prefill stage (*c*, *d*). **(a, c) Static Heads**: focus stably on a small set of tokens during decoding (*a*), consistent with high-sparsity text-centric attention (Eq. (2)) in prefill (*c*). **(b, d) Dynamic Heads**: shift focus across many tokens during decoding (*b*), consistent with low-sparsity distributions in prefill (*d*).

$\mathbb{I}(\cdot)$ is the indicator function for counting, and $k = \lceil 0.05 \times C \rceil$ with C the context length. We set $N = 500$ to track behavior across a long generation.

Analyzing the cumulative focus counts reveals two clear patterns: (i) **static pattern**, a head repeatedly attends to a specific and limited set of context tokens (see Fig. 2 (a)), indicating stable focus; and (ii) **dynamic pattern**, a head distributes attention more fluidly across many tokens (see Fig. 2 (b)), indicating evolving focus.

To explain these head patterns, we further analyze the prefill stage, when the MLLM forms cross-modal understanding by attending from the text prompt to the entire context. We employ ***text-centric attention*** to quantify this interaction:

$$\mathbf{S}_{\text{text}} = \text{Softmax}\left(\frac{\mathbf{Q}_{\text{text}}\mathbf{K}_{\text{context}}^{\top}}{\sqrt{d_k}}\right), \qquad (2)$$

where $\mathbf{Q}_{\text{text}} \in \mathbb{R}^{T\times d}$ are the vectors of query state derived from the T input prompt text tokens, $\mathbf{K}_{\text{context}} \in \mathbb{R}^{C\times d}$ are the key vectors of the entire C context tokens, and d is the hidden state dimension. Examining the sparsity of the matrix $\mathbf{S}_{\text{text}}$, we find that heads also exhibit two distinct patterns in prefill: **high-sparsity** (Fig. 2 (c)), where most attention focuses on a few key tokens, and **low-sparsity** (Fig. 2 (d)), where attention spreads broadly.

Our analysis uncovers a *clear* predictive link between prefill and decode behaviors. Heads with high-sparsity text-centric attention in the prefill stage consistently exhibit a static pattern in the decode stage (Fig. 2 (a, c)); while those with low-sparsity attention align with the dynamic pattern (Fig. 2 (b, d)). The intuition is straightforward: when a head, guided by the text prompt, concentrates on a small set of tokens early on, its focus stabilizes during decoding. Conversely, a diffuse prefill distribution implies that many tokens may become relevant later, prompting the head to shift attention dynamically. ➠ ***Thus, the sparsity of text-centric attention in the prefill stage provides a reliable and efficient proxy for predicting whether attention heads behave statically or dynamically during decoding.***

2.2 Variation of Patterns Across Tasks

Section 2.1 has shown that prefill sparsity predicts whether an attention head follows a static or dynamic pattern. A natural follow-up question is whether a head's such behavior is consistent across tasks or varies with context. To examine this, we evaluate `Qwen2.5-VL-7B` on multiple video and image benchmarks. We use ***cumulative attention distribution*** curves to quantify the head's attention sparsity during the prefill stage, as shown in Fig. 3. The results reveal clear *task-dependent variation*. For example, in Fig. 3 (a), the same head shows a highly sparse, static pattern on VATEX (blue solid line: top 5% tokens cover $>$ 95% of attention), but a much denser, dynamic pattern on NextQA (blue dashed line: top 5% tokens cover $<$ 75% of attention). ➠ ***Thus, a head can behave statically in one task while dynamically in another.***

Even within the static family (say, heads with $>$ 90% cumulative attention), sparsity levels vary. In Fig. 3 (b, c), two heads from SlideVQA both appear static, yet one focuses nearly 95% of its mass on the top 5% tokens, while the other barely crosses 90%. A similar pattern is observed in VATEX (Fig. 3(a, d)). ➠ ***Hence, heads not only switch between static and dynamic roles across tasks but also vary in strength within the same role.***

Since prefill sparsity governs decoding behavior, these differences confirm that a head's role is task-

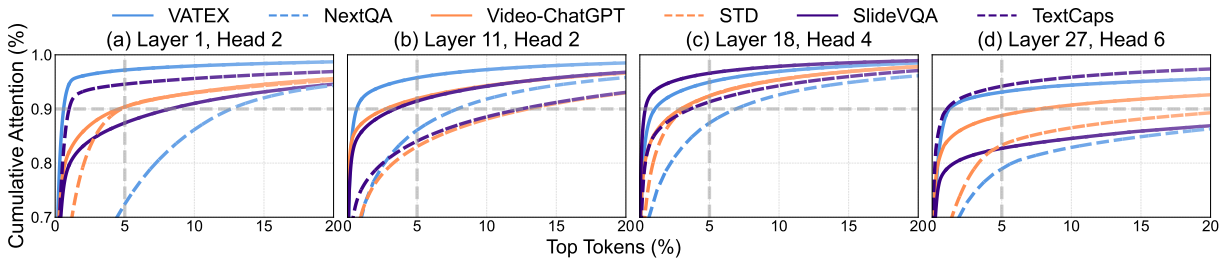


Figure 3: Cumulative attention distribution during prefill for selected heads in `Qwen2.5-VL-7B`. Each curve shows the fraction of attention captured by the top $x\%$ tokens across video tasks (VATEX, NextQA, Video-ChatGPT) and image tasks (STD, SlideVQA, TextCaps).

dependent. Effective compression must therefore be context-aware, *adapting to both input and task*.

3 HybridKV

Given the insights, we present HybridKV, a hybrid KV cache compression framework for efficient yet effective MLLM inference (Fig. 4). HybridKV exploits the heterogeneity of attention heads with a context-aware, head-level design. It first classifies heads into static or dynamic based on text-centric sparsity (Section 3.1), then applies a top-down budget allocation to distribute cache resources hierarchically (Section 3.2). Finally, it integrates static pruning with dynamic retrieval into a unified hybrid compression strategy (Section 3.3).

3.1 Text-centric Head Classification

As noted in Section 2.1, MLLM attention heads exhibit sparsity in text-centric attention (Eq. (2)), with most attention focused on a small subset of tokens in the entire context. Static heads show sharper focus on these tokens, while dynamic heads attend more broadly. This contrast motivates a principled metric for head classification.

Building on Section 2.2, this metric should be *context-aware*, adapting to different inputs and tasks to ensure accurate classification across diverse conditions. To this end, we propose the ***text-centric sparsity score***, which separates static and dynamic heads during the prefill stage. The score is computed in two steps: (i) *intra-token*: retain only the top-k positions from each text token's distribution to suppress long-tail noise; (ii) *inter-token*: aggregate these filtered values into a head-level profile. Formally, for the h-th head in layer l with text token set T:

$$S_{l,h} = \frac{1}{|T|} \sum_{i=1}^{|T|} \sum_{j=1}^{k} \mathrm{TopK}(\mathbf{S}_{\text{text}}[i], k), \quad (3)$$

where $\mathbf{S}_{\text{text}}[i]$ denotes the attention distribution over all tokens when attending from text token i, and $\mathrm{TopK}(\cdot, k)$ selects the top-k entries. Finally, we classify heads by thresholding $S_{l,h}$ against a quantile-based cutoff θ tuned on a validation set. Heads with $S_{l,h} \geq \theta$ are labeled *static*, while those with $S_{l,h} < \theta$ are labels *dynamic* (see Section 4.5 for a sensitivity study on θ). Based on this classification, we can apply a hybrid compression strategy (Section 3.3): Static heads exhibit stable attention and are compressed more aggressively via pruning, while dynamic heads adapt their focus during decoding and we use retrieval to fetch relevant tokens at runtime. As pruning and retrieval entail different efficiency–accuracy trade-offs, we develop a tailored budget allocation to balance the two.

3.2 Top-down Budget Allocation

Existing KV cache compression (Wan et al., 2024b; Li et al., 2025; Wang et al., 2025) for MLLMs mainly relies on uniform pruning across heads, disregarding their distinct roles and behaviors. In contrast, our method leverages head classification to perform hierarchical budget allocation, first distributing *budgets between head types*, then refining them *across individual heads*, achieving a balanced trade-off between generation quality and efficiency.

Budget Allocation at the Head-type Level. This step is critical as an imbalanced allocation can degrade both accuracy and efficiency. Over-allocating to static heads limits the capacity to track evolving focus tokens of dynamic heads during decoding, whereas excessive allocation to dynamic heads discards important prefilling tokens and increases KV cache I/O overhead from dynamic retrieval (as depicted in Section 3.3). To balance the two, we allocate the total KV budget B_{total} across N_{stat} static and N_{dyna} dynamic heads using the average per-head budget $\bar{B} = B_{\text{total}}/(N_{\text{stat}} + N_{\text{dyna}})$. A share coefficient r, constrained by not exceeding total budget such that $r \leq B_{\text{total}}/(\bar{B} \cdot N_{\text{dyna}})$, con-

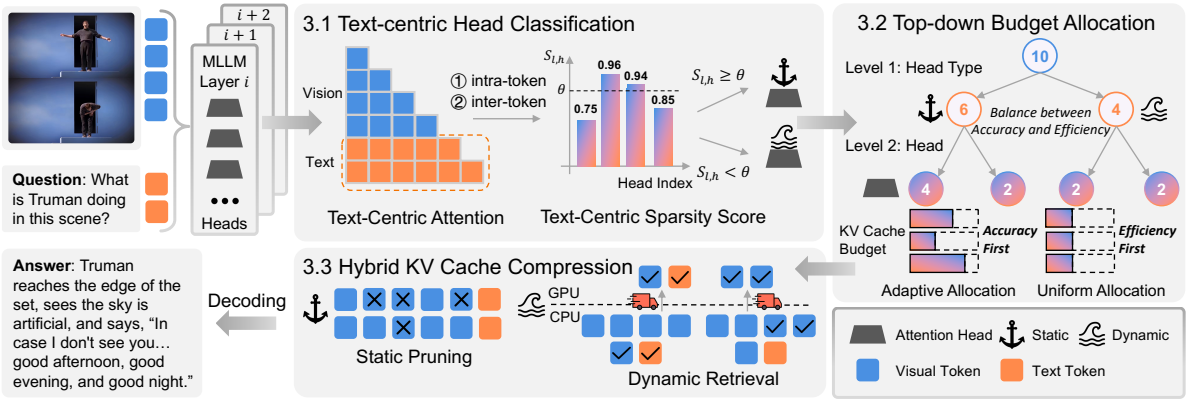


Figure 4: Overview of HybridKV. Section 3.1: Heads are classified by text-centric sparsity into static (anchor, few tokens) and dynamic (wave, broad focus). Section 3.2: A two-layer scheme allocates KV cache budgets first by head type and individual heads then. Section 3.3: Hybrid compression integrates static pruning (drop uninformative tokens) with dynamic retrieval (reactivate tokens when needed) to optimize cache usage.

trols the portion assigned to dynamic heads:

$$B_{\text{dyna}} = \lceil r \cdot \bar{B} \cdot N_{\text{dyna}} \rceil;\ B_{\text{stat}} = B_{\text{total}} - B_{\text{dyna}}. \quad (4)$$

The coefficient r modulates the accuracy–efficiency trade-off, with $r = 1$ indicating uniform allocation for the two head types. Its sensitivity analysis is provided in Section 4.5.

Budget Allocation at the Individual Head Level. After determining type-level budgets, we further distribute them among individual heads while satisfying total memory constraints.

For static heads, whose importance varies across tasks (as observed in Section 2.2), we allocate budgets adaptively using the text-centric sparsity score $S_{l,h}$ from Section 3.1. As this score is calculated for the current input, it is inherently task-specific to quantify the importance of each head. The scores are normalized across all static heads with L_1 normalization. Formally, we denote the set of static heads in layer l as $\mathcal{H}_{\text{stat},l}$. If the h-th head in layer l is in $\mathcal{H}_{\text{stat},l}$, the budget for this static head is allocated as the sum of a uniform base budget and a score-proportional budget. We define $\alpha \in (0, 1)$ as the allocation ratio, controlling the balance between the two types of budget. Formally, we compute each head's budget as

$$b_{l,h} = \lceil \underbrace{\alpha \cdot B_{\text{stat}} \cdot N_{\text{stat}}^{-1}}_{\text{Base Budget}} + \underbrace{(1-\alpha) \cdot B_{\text{stat}} \cdot S_{l,h}}_{\text{Proportional Budget}} \rceil, \quad (5)$$

This adaptive strategy ensures that important static heads receive sufficient resources, improving accuracy across diverse multimodal tasks.

For dynamic heads, to be compatible with the efficient implementation of the dynamic retrieval technique introduced in Section 3.3, we simply allocate the total budget B_{dyna} uniformly across N_{dyna} dynamic heads, consistent with previous studies (Tang et al., 2024; Sun et al., 2024). The assigned budget is padded to its nearest powers of 2 to accommodate CUDA constraints in dynamic KV cache handling. This strategy maintains decoding efficiency with minimal accuracy loss.

3.3 Hybrid KV Cache Compression

Building upon head classification and top-down budget allocation, we implement a hybrid KV cache compression process. For static heads, whose attention targets stable text and visual positions, we apply ***text-prior static pruning***: during prefill, KV caches for text tokens are retained with higher priority, followed by those for local and salient visual tokens, reducing memory usage. For dynamic heads, which attend to small and variable subsets of multimodal inputs, we adopt ***chunk-wise dynamic retrieval***, widely used in recent KV cache methods (Tang et al., 2024; Sun et al., 2024). KV caches are offloaded to the CPU to satisfy memory constraints, and a chunk-level index is built for efficient retrieval in prefill. Important KV caches are selected dynamically for attention computation during decoding.

A unified GPU KV cache buffer accommodates important tokens from both head types, filled by static heads during prefill and updated by dynamic heads during decoding. Implementation details are provided in Section A. Importantly, our hybrid compression process is orthogonal to existing static pruning and dynamic retrieval methods (see Section B) and can be readily combined with them for further gains in accuracy and efficiency.

4 Experiments

Baselines. We compare against four recent KV cache compression methods for MLLMs. ① SnapKV (Li et al., 2024b) and ② LOOK-M (Wan et al., 2024b) allocate budgets at the token level: SnapKV retains crucial tokens using cumulative attention scores, while LOOK-M prioritizes text tokens and applies KV cache merging for MLLMs; ③ MadaKV (Li et al., 2025) distributes budgets across layers based on a modality-preference metric; ④ SparseMM (Wang et al., 2025) allocates proportional budgets to heads using offline visual head scores. All these emphasize budget allocation without considering compression strategies tailored to different head types.

Tasks and Models. We evaluate HybridKV on both image and video tasks using MileBench (Song et al., 2024) and LMMs-Eval (Zhang et al., 2025b). Detailed benchmark information is provided in Section C. The MLLMs include `Qwen2.5-VL-3B`, `Qwen2.5-VL-7B`, and `LLaVA-OneVision-7B` [2]. For video tasks, we sample 64 frames per video input. All experiments are run on NVIDIA L40S GPUs.

4.1 Performance Results

The overall comparison results on the image and video tasks are reported in Table 1 and Table 2, respectively. We allocate only 10% of the full KV cache to evaluate performance under high-compression settings. Despite this strict constraint, HybridKV maintains accuracy comparable to, and sometimes even surpasses, the Full Cache baseline (gray bars) on different model sizes and architectures, yielding greater performance gains on larger models. These results demonstrate that HybridKV substantially reduces the memory footprint with a negligible impact on performance. Such savings are crucial for deploying MLLMs in memory-intensive scenarios like long video understanding (e.g., VATEX) and multi-image reasoning (e.g., MM-QA), where the uncompressed KV caches exceed the capacity of even high-end GPUs.

Image Tasks. As shown in Table 1, HybridKV yields accuracy comparable to the Full Cache baseline (within 1.3% drop in average) across eight tasks included in MileBench (Song et al., 2024), which cover complex multi-image settings. It outperforms SnapKV, LOOK-M and MadaKV, which rely on token- and layer-level budget allocation, across almost all MLLMs and datasets, showing the effectiveness of our head-level compression strategy. It also surpasses SparseMM, which adopt unified pruning with head-level budget allocation, indicating the necessity of analyzing head properties and classifying heads accordingly in HybridKV's design. These results highlight the advantage of our hybrid compression scheme, which integrates tailored strategies with adaptive budget allocation for different types of attention heads in MLLMs.

Video Tasks. We further evaluate HybridKV on three video tasks, with results summarized in Table 2. Compared with image input, video input contains temporal information between frames, which poses more challenges to the design of KV cache compression. Even in this difficult setting, HybridKV achieves higher accuracy across most metrics, while baseline methods struggle to deliver consistent performance. For instance, SparseMM performs well on certain benchmarks (NextQA) but suffers notable degradation on others (5% average accuracy drop in VATEX) since it only captures head properties in an offline manner. In contrast, HybridKV even exceeds the Full Cache baseline in VATEX by keeping the most salient subset of tokens during decoding, not only preserving but also enhancing the model's capabilities. This demonstrates the effectiveness of our context-aware design in retaining important information within KV caches for different tasks.

4.2 Efficiency Results

We assess the efficiency of HybridKV on `Qwen2.5-VL-7B` using Video-ChatGPT (Maaz et al., 2024) for real-world long video understanding scenarios. We randomly sample 20 data entries and set the maximum generation length to 128 tokens for evaluation. All experiments use FlashAttention (Dao, 2023). Table 3 shows HybridKV markedly reduces both GPU memory and decoding latency relative to the Full Cache baseline. With a 20% cache budget, HybridKV results in 4.3× KV cache memory savings and faster decoding; at 10%, latency further decreases to 38.65 ms-per-token with only 0.22 GB memory, achieving 1.52× decoding speedup and 7.9× KV cache GPU memory reduction compared to Full Cache. These efficiency gains show that HybridKV is well-suited for scaling MLLMs to long video and multi-image inputs. A lower budget enables higher memory utilization and faster decoding. Both 10% and 20% are effective compression ratios, with 10% offering greater deployment flexibility when a minor quality

[2] We exclude evaluating `InternVL` as SparseMM does not provide its visual head score and lacks support for it.

	Method	CL-CH	DocVQA	MMCoQA	MM-QA	SlideVQA	STD	WebQA	WikiVQA	Average
Qwen2.5-VL-7B	FULL CACHE	43.07	97.50	66.50	75.00	83.50	29.79	76.50	92.50	70.55
	SNAPKV	37.41	97.50	54.00	73.00	83.00	29.02	71.00	89.50	66.80
	LOOK-M	38.81	97.50	52.50	73.50	82.50	29.25	71.00	88.00	66.63
	MADAKV	37.86	97.50	55.00	74.50	82.00	28.24	70.50	90.00	66.95
	SPARSEMM	37.63	97.50	62.00	75.50	83.50	28.69	70.50	92.50	68.48
	HYBRIDKV	**40.57**	**97.50**	**63.00**	**76.00**	**83.50**	**29.84**	**76.00**	**93.00**	**69.93**
Qwen2.5-VL-3B	FULL CACHE	38.34	96.00	56.50	80.00	80.00	29.37	72.00	88.00	67.53
	SNAPKV	34.78	96.00	43.00	78.50	80.00	26.16	68.00	83.50	63.74
	LOOK-M	35.12	96.00	42.50	79.00	80.00	26.24	66.50	83.50	63.61
	MADAKV	33.66	96.00	43.50	79.50	80.00	26.02	66.00	84.00	63.59
	SPARSEMM	35.05	96.00	50.50	79.00	79.50	26.80	65.00	87.00	64.86
	HYBRIDKV	**37.75**	**96.00**	**51.00**	**79.50**	**80.00**	**29.22**	**72.00**	**88.00**	**66.68**
LLaVA-OV-7B	FULL CACHE	44.76	97.00	53.50	77.00	77.50	31.23	76.50	88.00	68.18
	SNAPKV	44.13	97.00	51.50	77.00	77.50	30.17	76.50	87.50	67.66
	LOOK-M	44.15	97.00	51.00	77.00	77.50	30.57	76.50	87.50	67.65
	MADAKV	43.37	97.00	52.00	77.00	77.50	30.69	76.50	87.50	67.70
	SPARSEMM	43.72	97.00	52.50	77.00	77.50	30.56	76.50	86.50	67.66
	HYBRIDKV	**44.31**	**97.00**	**52.50**	**77.00**	**77.50**	**30.71**	**76.50**	**88.00**	**67.94**

Table 1: Performance of five KV cache compression strategies on three MLLMs on image tasks. ROUGE-L (Lin, 2004) evaluates generated–reference consistency for CL-CH and STD, while *exact match accuracy* is used for other QA tasks. Higher values indicate better performance, and the best measures are highlighted in **bold** by default.

	Method	VATEX					NextQA	Video-ChatGPT					
		BLEU-4	Meteor	ROUGE-L	CIDEr	Average	WUPS	CI	DO	CU	TU	CO	Average
Qwen2.5-VL-7B	FULL CACHE	0.2181	0.2209	0.4266	0.4628	0.3321	33.79	3.06	3.15	3.52	2.24	2.69	2.93
	SNAPKV	0.2146	0.2125	0.4240	0.4498	0.3252	33.29	2.90	2.96	3.40	1.95	2.53	2.75
	LOOK-M	0.2180	0.2174	0.4280	0.4571	0.3301	33.07	2.92	2.97	3.38	2.05	2.49	2.76
	MADAKV	0.2051	0.2157	0.4228	0.4306	0.3186	33.25	2.94	3.03	3.41	2.02	2.56	2.79
	SPARSEMM	0.2133	0.2116	0.4227	0.4198	0.3169	33.84	2.90	2.95	3.37	1.99	2.52	2.75
	HYBRIDKV	**0.2266**	**0.2236**	**0.4318**	**0.4774**	**0.3399**	**33.86**	**2.99**	**3.05**	**3.47**	**2.15**	**2.57**	**2.85**
Qwen2.5-VL-3B	FULL CACHE	0.2979	0.2330	0.4896	0.5661	0.3967	29.81	2.59	2.67	3.06	1.93	2.49	2.55
	SNAPKV	0.2742	0.2186	0.4748	0.5072	0.3687	30.10	2.64	2.67	3.09	1.77	2.23	2.48
	LOOK-M	0.2851	0.2245	0.4756	0.5176	0.3757	30.49	**2.74**	2.68	3.10	**1.85**	2.23	2.52
	MADAKV	0.2691	0.2095	0.4597	0.4708	0.3523	30.00	2.60	2.64	3.03	1.81	2.39	2.49
	SPARSEMM	0.2708	0.2080	0.4544	0.4352	0.3421	**31.07**	2.69	2.66	3.11	1.82	2.25	2.51
	HYBRIDKV	**0.2936**	**0.2286**	**0.4772**	**0.5427**	**0.3855**	30.70	2.70	**2.72**	**3.11**	1.84	**2.43**	**2.56**
LLaVA-OV-7B	FULL CACHE	0.1330	0.1900	0.3739	0.2355	0.2331	20.18	2.99	2.86	3.36	2.00	2.99	2.84
	SNAPKV	0.1161	0.1704	0.3584	0.1799	0.2062	18.56	2.88	2.78	3.31	1.83	2.80	2.72
	LOOK-M	0.1205	0.1772	0.3625	0.1957	0.2140	18.69	2.90	2.75	3.32	1.87	2.87	2.74
	MADAKV	0.1115	0.1748	0.3601	0.1744	0.2052	18.79	2.90	2.79	3.35	1.90	2.89	2.77
	SPARSEMM	0.1193	0.1688	0.3593	0.1770	0.2061	18.86	2.87	2.73	3.27	**1.91**	2.84	2.72
	HYBRIDKV	**0.1374**	**0.1800**	**0.3700**	**0.2109**	**0.2246**	**19.62**	**2.96**	**2.88**	**3.35**	1.89	**2.89**	**2.79**

Table 2: Performance of five KV cache compression strategies on three MLLMs on video tasks. Evaluation metrics (e.g., BLEU-4 and Meteor) are explained in Section C; generally, higher measures indicate better performance.

Method	Budget	Accuracy (avg.)	GPU Memory (GB)	Latency (ms/token)
FULL CACHE	100%	2.93	1.73	58.94
HYBRIDKV	20%	**2.88**	0.40	42.08
HYBRIDKV	10%	2.85	**0.22**	**38.65**

Table 3: KV cache GPU memory usage and decoding latency on `Qwen2.5-VL-7B` with Video-ChatGPT.

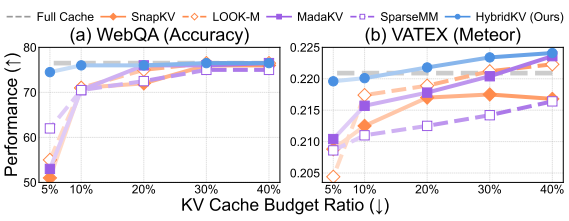


Figure 5: Scalability under different KV cache budgets on WebQA and VATEX. HYBRIDKV outperforms all baselines across budgets, matching or exceeding full cache performance and maintaining high accuracy even under aggressive compression (e.g., 5% budget).

trade-off is acceptable.

4.3 Scaling Law for KV Cache Budgets

To investigate the scalability of KV cache compression strategies, we conduct experiments on `Qwen2.5-VL-7B` with cache budgets ranging from 5% to 40% on WebQA and VATEX. This setting covers both multi-image QA and video captioning tasks. As shown in Fig. 5, HYBRIDKV consistently outperforms all baselines across all budget ratios. Its performance closely tracks the FULL CACHE baseline and even surpasses it on VATEX, indicating that HYBRIDKV effectively filters out noisy tokens and sharpens focus on salient visual regions. The advantage is most pronounced under extreme compression (<10% budget), where baselines degrade sharply but HYBRIDKV maintains strong accuracy. These results highlight that HYBRIDKV scales effectively by preserving critical information in KV caches, thereby reducing memory cost with minimal performance loss.

Method	STD	WebQA	NextQA	Latency (ms/token)
HybridKV	**29.84**	**76.00**	**33.86**	38.65
w. all static heads	28.72	73.00	33.61	**34.13**
w. all dynamic heads	29.77	75.50	33.76	48.15

Table 4: Effect of head classification.

θ	CL-CH	STD	WebQA	Average
0.75	39.06	29.53	72.50	47.03
0.80	39.49	29.41	72.50	47.13
0.85	39.87	29.51	75.00	48.13
0.90	**40.57**	**29.84**	**76.00**	**48.83**
0.95	39.13	29.69	76.00	48.27

Table 5: Sensitivity study on the score threshold θ. For this analysis, the share coefficient is fixed at $r = 0.75$ and the KV cache budget is 10%.

4.4 Ablation Study

Here, we analyze the effect of each component in HybridKV through ablations on head classification and budget allocation. Results are reported on three subtasks, Spot-the-Diff (STD), WebQA for image tasks, and NextQA for video tasks, with latency measured under a 10% KV cache budget.

Effect of Head Classification. Table 4 compares HybridKV with two variants that assign all heads as either static or dynamic under the same budget allocation strategy. Both variants suffer accuracy degradation across tasks, since treating all heads identically overlooks their heterogeneous patterns in MLLMs. The variant with all dynamic heads results in higher latency due to more KV cache I/O overhead. In contrast, HybridKV leverages text-centric classification to adaptively identify head properties from input contexts, yielding better accuracy without significant latency overhead.

Effect of Budget Allocation. Table 7 compares HybridKV with two variants fixing the same head classification process: one that removes head-level adaptive allocation for static heads, and a uniform baseline that ignores both head type and individual head. HybridKV exhibits better accuracy on both image and video tasks while maintaining decoding latency, showing the robustness of our budget allocation. Removing head-level allocation (i.e. w/o H) degrades accuracy due to ignoring the variable importance of individual heads. When head-type allocation is also removed (i.e. w/o (H & HT)), accuracy declines further because tailored strategies for static and dynamic heads are ignored, leading to imbalanced budgets. This also increases decoding latency due to the over-allocation to dynamic heads. These results confirm that our top-down allocation, combining head-type and head-level granularity, improves both accuracy and efficiency and complements the hybrid compression design.

4.5 Sensitivity Study on Hyperparameters

We conduct a sensitivity analysis to validate the choices of our key hyperparmeters, score threshold θ and share coefficient r. All experiments are performed on the `Qwen2.5-VL-7B` model across three datasets: CLEVR-Change (CL-CH), Spot-the-Diff (STD), and WebQA. When analyzing one hyperparameter, all other parameters are held constant at their default optimal values.

Score Threshold θ in Head Classification. The score threshold θ is a critical hyperparameter in our text-centric head classification process, acting as the decision boundary for categorizing an attention head as static or dynamic. The design of θ is based on the intrinsic property of attention sparsity, which results in the scores for most heads being clustered at the higher end of the spectrum. Consequently, any meaningful decision boundary for separating static from dynamic heads must also reside in a high-value range. Thus, our search is focused on the high-score interval $[0.75, 0.95]$. The results, summarized in Table 5, show that the choice of θ directly impacts model performance. A threshold set too low, such as $\theta = 0.75$ may incorrectly classify heads with dynamic properties as static. This prevents these heads' recall of critical tokens from the context, leading to a loss of essential information and causing a drop of 1.8 points in average accuracy compared to the peak performance. Conversely, an excessively high threshold like $\theta = 0.95$ creates too many dynamic heads. This not only increases data transmission overhead during inference due to the costly retrieval operations, but also heightens the risk of introducing noise from irrelevant tokens, resulting in a slight dip in performance. Therefore, **we select $\theta = 0.9$ as it achieves the best accuracy while maintaining a reasonable number of dynamic heads for efficiency.**

Share Coefficient r in Budget Allocation. In our top-down budget allocation process, the share coefficient r controls the proportion of the KV cache for two types of head. Our search for its optimal value is designed to cover several operational regimes to fully characterize its impact: values of $r \ll 1.0$ to investigate a resource starvation scenario for the dynamic heads, values of $r \approx 1.0$ to access a balanced allocation, and val-

r	CL-CH	STD	WebQA	Average
0.1	39.26	29.37	75.00	47.88
0.25	39.87	29.65	75.50	48.34
0.50	40.03	29.75	75.50	48.43
0.75	**40.57**	**29.84**	**76.00**	**48.83**
1.00	40.12	29.68	74.00	47.93
1.25	39.02	29.44	74.50	47.65

Table 6: Sensitivity study on the share coefficient r. For this analysis, the score threshold is fixed at $\theta = 0.90$ and the KV cache budget is 10%.

Method	STD	WebQA	NextQA	Latency (ms/token)
HybridKV	**29.84**	**76.00**	**33.86**	**38.65**
w/o H	29.67	76.00	33.70	38.65
w/o (H & HT)	29.58	74.50	33.29	44.67

Table 7: Effect of budget allocation. H: head-level budget allocation; HT: head-type-level budget allocation.

ues of $r > 1.0$ to observe the effects of favoring dynamic heads. Notably, r is naturally constrained by the total budget and must satisfy the condition $r \leq B_{\text{total}}/(\bar{B} * N_{\text{dyna}})$ to avoid exceeding available cache budget. The results in Table 6 clearly illustrate the behavior across these regimes. A coefficient set too low provides insufficient budget for the dynamic heads, harming their ability to retrieve important tokens and degrading model accuracy. Conversely, a coefficient set too high is also suboptimal. It allocates an excessive budget to the dynamic heads, which increase data transmission overhead without any meaningful gain in accuracy, while simultaneously starving the static heads of their required resources. This imbalance ultimately harms collective performance. This demonstrates the need for a moderate, well-balanced coefficient. Therefore, **we select $r = 0.75$ as the optimal choice as it achieves the highest empirical accuracy. This moderate value strikes the ideal balance, providing a relatively sufficient and balanced budget for both types of head, while avoiding the increased data transmission overhead and budget overflow risk associated with higher coefficients.**

5 Related Work

Existing KV cache compression methods in LLMs fall into *static pruning* and *dynamic retrieval*. Static methods (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024b) discard KV caches in the prefill stage using fixed criteria, while dynamic methods (Tang et al., 2024; Sun et al., 2024; Chen et al., 2024b; Yao et al., 2025) retrieve important KV caches during decoding. These paradigms extend to MLLMs with compression strategies at the token-level (Wan et al., 2024b), layer-level (Li et al., 2025; Wan et al., 2025), and head-level (Wang et al., 2025). However, a common limitation is that they apply either a fully static or a fully dynamic strategy, thus overlooking the diverse patterns of attention heads. In contrast, HybridKV is founded on the observation of heterogeneous attention patterns and introduces a novel hybrid KV cache compression framework for efficient MLLM inference. We classify heads into static and dynamic types using text-centric attention embedded in multimodal inputs, and apply a tailored budget allocation and compression strategy to each.

Another line of work directly addresses the redundancy of visual information by pruning or merging visual tokens, either in the vision encoder (Yang et al., 2025) or within the language model (Chen et al., 2024a; Zhang et al., 2025c; Hu et al., 2025). These methods rely on attention scores or frame uniqueness (Liu et al., 2025b) to identify and discard less important tokens. Differently, our method tackles visual redundancy from the heterogeneity of attention heads in MLLMs, providing a more fundamental and effective mechanism for managing visual information within the KV cache. Section B provides an extended discussion of related work.

6 Conclusion

We present HybridKV, a hybrid KV cache compression framework for efficient MLLM inference. By identifying distinct attention head patterns and applying hierarchical, head-aware compression, HybridKV cuts KV cache GPU memory by 7.9× and speeds up decoding by 1.52×, all while preserving generation quality on `Qwen2.5-VL-7B`. We envision HybridKV as both a valuable community tool for deployment-ready MLLMs and a foundation for future strategy-driven, context-aware compression in multimodal reasoning.

7 Acknowledgements

The work was supported by the Major Research Program of the Zhejiang Provincial Natural Science Foundation (Grant No. LD24F020015), National Natural Science Foundation of China (Grant No. 62471168), CCF-Baidu Open Fund (No. 202509), and Zhejiang Province "Leading Talent of Technological Innovation Program" (No. 2023R5214).

8 Ethical Considerations

All experiments in this work are conducted using open-source datasets and models. Our research focuses solely on improving inference efficiency and does not involve any sensitive data, human subjects, or commercial use [3] [4] [5] [6].

9 Limitation

While HYBRIDKV delivers strong improvements in memory efficiency and decoding speed, it also has certain limitations. First, our head classification relies on text-centric attention patterns observed during the prefill stage. Although we show this to be a reliable proxy, the classification depends on thresholding and may vary across model scales or domains. More sophisticated classifiers (e.g., learned or adaptive) could further improve robustness but are beyond the focus of this work. Second, the current design targets inference-time efficiency without involving additional training or fine-tuning (Zhang et al., 2025a). This makes HYBRIDKV broadly applicable, but it also limits opportunities to co-adapt compression strategies with model training. Third, our evaluation primarily considers image and video benchmarks where KV cache growth is most severe. Extending to other modalities and tasks (e.g., audio-grounded MLLMs or extremely long-text reasoning) could reveal additional challenges. Fourth, our text-centric head classification is tailored to the distinct behaviors of MLLMs, making direct application to text-only LLMs suboptimal.

Nevertheless, HYBRIDKV's core principle of exploiting head-level heterogeneity is highly transferable. Our framework is compatible with future extensions such as adaptive thresholding, joint training, or integration with other deployment optimizations (e.g., quantization or speculative decoding (Ji et al., 2025, 2026)). Furthermore, adapting this paradigm for text-only models through alternative classification schemes (e.g., attention entropy, consistent local patterns, or learned gating) represents a promising future direction. We believe these efforts will continually enhance the generality and scalability of HYBRIDKV.

[3] https://github.com/MileBench/MileBench (Apache License 2.0)

[4] https://huggingface.co/datasets/lmms-lab/VATEX (Apache License 2.0)

[5] https://huggingface.co/datasets/lmms-lab/NExTQA (Apache License 2.0)

[6] https://huggingface.co/datasets/lmms-lab/VideoChatGPT (Apache License 2.0)

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and 1 others. 2024. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. 2022. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16495–16504.

Junjie Chen, Xuyang Liu, Zichen Wen, Yiyu Wang, Siteng Huang, and Honggang Chen. 2025a. Variation-aware vision token dropping for faster large vision-language models. *arXiv preprint arXiv:2509.01552*.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024a. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer.

Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. 2024b. Arkvale: Efficient generative llm inference with recallable key-value eviction. *Advances in Neural Information Processing Systems*, 37:113134–113155.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, and 1 others. 2024c. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and 1 others. 2025b. Magicpig: Lsh sampling for efficient llm generation. In *The Thirteenth International Conference on Learning Representations*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Yuan Feng, Haoyu Guo, JunLin Lv, S. Kevin Zhou, and Xike Xie. 2025a. Taming the fragility of kv cache eviction in llm inference. *Preprint*, arXiv:2510.13334.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025b. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025c. Identify critical kv cache in llm inference from an output perturbation perspective. *Preprint*, arXiv:2502.03805.

Yuhang Han, Xuyang Liu, Zihan Zhang, Pengxiang Ding, Junjie Chen, Honggang Chen, Donglin Wang, Qingsen Yan, and Siteng Huang. 2026. Filter, correlate, compress: Training-free token reduction for mllm acceleration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 4601–4609.

Lianyu Hu, Fanhua Shang, Wei Feng, and Liang Wan. 2025. Lightvlm: Acceleraing large multimodal models with pyramid token merging and kv cache compression. *arXiv preprint arXiv:2509.00419*.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. 2018. Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4024–4034.

Yicheng Ji, Jun Zhang, Jinpeng Chen, Cong Wang, Lidan Shou, Gang Chen, and Huan Li. 2026. See the forest for the trees: Loosely speculative decoding via visual-semantic guidance for efficient inference of video llms. *Preprint*, arXiv:2604.05650.

Yicheng Ji, Jun Zhang, Heming Xia, Jinpeng Chen, Lidan Shou, Gang Chen, and Huan Li. 2025. SpecVLM: Enhancing Speculative Decoding of Video LLMs via Verifier-Guided Token Pruning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7216–7230.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024a. Llava-onevision: Easy visual task transfer. *CoRR*.

Kunxi Li, Zhonghua Jiang, Zhouzhou Shen, ZhaodeWang ZhaodeWang, Chengfei Lv, Shengyu Zhang, Fan Wu, and Fei Wu. 2025. Madakv: Adaptive modality-perception kv cache eviction for efficient multimodal long-context inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13306–13318.

Yongqi Li, Wenjie Li, and Liqiang Nie. 2022. Mmcoqa: Conversational question answering over text, tables, and images. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4220–4231.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Xinying Lin, Xuyang Liu, Yiyu Wang, Teng Ma, and Wenqi Ren. 2026. V-cast: Video curvature-aware spatio-temporal pruning for efficient video large language models. *arXiv preprint arXiv:2603.27650*.

Xuyang Liu, Xiyan Gui, Yuchao Zhang, and Linfeng Zhang. 2025a. Mixing importance with diversity: Joint optimization for kv cache compression in large vision-language models. *arXiv preprint arXiv:2510.20707*.

Xuyang Liu, Yiyu Wang, Junpeng Ma, and Linfeng Zhang. 2025b. Video compression commander: Plug-and-play inference acceleration for video large language models. *arXiv preprint arXiv:2505.14454*.

Xuyang Liu, Ziming Wang, Junjie Chen, Yuhang Han, Yingyao Wang, Jiale Yuan, Jun Song, Siteng Huang, and Honggang Chen. 2026. Global compression commander: Plug-and-play inference acceleration for high-resolution large vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 7350–7358.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. 2024. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12585–12602.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. *Advances in neural information processing systems*, 27.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Dong Huk Park, Trevor Darrell, and Anna Rohrbach. 2019. Robust change captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4624–4633.

Dingjie Song, Shunian Chen, Guiming Hardy Chen, Fei Yu, Xiang Wan, and Benyou Wang. 2024. Milebench: Benchmarking mllms in long context. *arXiv preprint arXiv:2404.18532*.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. 2024. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: Complex question answering over text, tables and images. *arXiv preprint arXiv:2104.06039*.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. 2023. Slidevqa: a dataset for document visual question answering on multiple images. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, pages 13636–13645.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. 2025. Razorattention: Efficient kv cache compression through retrieval heads. In *ICLR*.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: query-aware sparsity for efficient long-context llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, pages 47901–47911.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. 2025. Dycoke: Dynamic compression of tokens for fast video large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18992–19001.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Zhongwei Wan, Hui Shen, Xin Wang, Che Liu, Zheda Mai, and Mi Zhang. 2025. 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)*, pages 2485–2497.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and 1 others. 2024a. D2o: Dynamic discriminative operations for efficient long-context inference of large language models. *arXiv preprint arXiv:2406.13035*.

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

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

Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. 2019. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4581–4591.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Shang Yang, Haotian Tang, Yao Fu, Song Han, and 1 others. 2025. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*.

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

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. 2021. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786.

Jing Xiong, Jianghan Shen, Fanghua Ye, Chaofan Tao, Zhongwei Wan, Jianqiao Lu, Xun Wu, Chuanyang Zheng, Zhijiang Guo, Min Yang, Lingpeng Kong, and Ngai Wong. 2025a. Uncomp: Can matrix entropy uncover sparsity? - A compressor design from an uncertainty-aware perspective. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 4179–4199. Association for Computational Linguistics.

Jing Xiong, Jianghan Shen, Chuanyang Zheng, Zhongwei Wan, Chenyang Zhao, Chiwun Yang, Fanghua Ye, Hongxia Yang, Lingpeng Kong, and Ngai Wong. 2025b. Parallelcomp: Parallel long-context compressor for length extrapolation. In *International Conference on Machine Learning*, pages 68998–69016. PMLR.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. 2025. Visionzip: Longer is better but not necessary in vision

language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19792–19802.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2013–2018.

Dingyu Yao, Bowen Shen, Zheng Lin, Wei Liu, Jian Luan, Bin Wang, and Weiping Wang. 2025. Tailorkv: A hybrid framework for long-context inference via tailored kv cache optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20340–20359.

Jun Zhang, Yicheng Ji, Feiyang Ren, Yihang Li, Bowen Zeng, Zonghao Chen, Ke Chen, Lidan Shou, Gang Chen, and Huan Li. 2026. Efficient inference for large vision-language models: Bottlenecks, techniques, and prospects. *Preprint*, arXiv:2604.05546.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Yang You, Guiming Xie, Xuejian Gong, and Kunlong Zhou. 2025a. Train Small, Infer Large: Memory-Efficient LoRA Training for Large Language Models. *The Thirteenth International Conference on Learning Representations*.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and 1 others. 2025b. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 881–916.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and 1 others. 2025c. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *Forty-second International Conference on Machine Learning*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

A Details of Hybrid KV Cache Compression

A detailed description of our hybrid KV cache compression implementation is provided in Fig. 6.

KV Cache Pruning for Static Heads. For static heads, we apply a selective pruning strategy to fit their allocated budget. Inspired by methods like SparseMM (Wang et al., 2025), our policy employs an observation window to efficiently score the importance of all tokens within the initial prompt.

The final condensed cache is constructed from three distinct sets of tokens: (i) the observation window itself, (ii) all text tokens from the historical context (tokens before the window), and (iii) the Top-M most relevant visual tokens from the historical context. To determine the relevance of historical visual tokens, we use the queries from the observation window to attend to the entire prompt. Given query states $\mathbf{Q}$ and key states $\mathbf{K}$, we compute vector $\mathbf{s}_w$ by the attention scores:

$$\mathbf{S}_w = \text{softmax}\left(\frac{\mathbf{Q}[C-w:C,:]\mathbf{K}^\top}{\sqrt{d_k}}\right) \quad (6)$$

$$\mathbf{s}_w = \frac{1}{w}\sum_{i=1}^{w}\mathbf{S}_w[i,:] \quad (7)$$

where w denotes the length of the observation window, and C is the length of the entire context. This vector $\mathbf{s}_w \in \mathbb{R}^C$ assigns a relevance score to every token in the prompt. We then select the Top-M historical visual tokens based on their corresponding scores in this vector. The number of visual tokens to keep, M, is set by the remaining budget after accounting for the observation window and historical text tokens. This policy guarantees the preservation of the most recent context, complete textual instruction, and the most significant historical visual features.

KV Cache Retrieval for Dynamic Heads. For dynamic heads, we adopt a chunk-wise retrieval strategy widely used in recent KV cache methods (Tang et al., 2024; Sun et al., 2024). Specifically, the token sequence is divided into fixed-size chunks (e.g., 8 tokens each), and each chunk is represented by metadata computed as the average of its key vectors. As illustrated in Fig. 6, during the prefill stage we build a chunk-level index from the metadata, offload dynamic-head KV caches to CPU memory, and retain only the index on GPU. At each decoding step, we identify relevant chunks by computing the inner product between the query vector and chunk metadata, load the selected chunks from CPU, and update the GPU KV buffer accordingly. The number of loaded chunks is determined by the per-head budget and chunk size. The final attention output is then computed together with the static KV cache. This strategy adaptively retrieves essential contexts while achieving efficiency through chunk-wise data transfer.

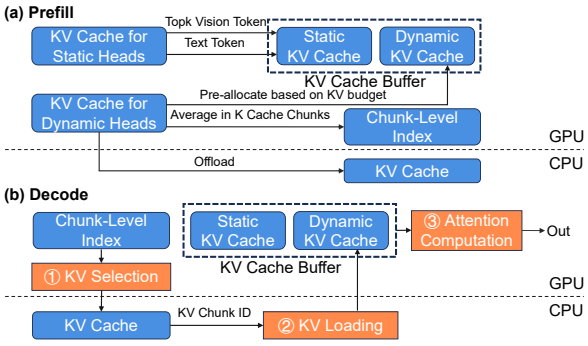


Figure 6: Illustration of hybrid KV cache compression.

B Details of Related Work

KV Cache Compression. In the LLM era, these methods fall into two categories: static pruning and dynamic retrieval. For static pruning methods, KV caches are discarded in the prefill stage. StreamingLLM (Xiao et al., 2024) retains attention sinks and recent tokens, while H2O (Zhang et al., 2023) and SnapKV (Li et al., 2024b) retain important tokens using cumulative attention scores. PyramidKV (Cai et al., 2024) allocates KV cache budgets across layers based on informativeness. Some methods focus on the head-level properties, DuoAttention (Xiao et al., 2025) and RazorAttention (Tang et al., 2025) divide attention heads into retrieval and non-retrieval categories, and store all tokens in retrieval heads while applying pruning for others. AdaKV (Feng et al., 2025b), on the other hand, proposes an adaptive budget allocation algorithm and can be integrated into existing methods. However, these one-time static pruning methods perform well in efficiency but often fail in model performance due to severe information loss. For dynamic retrieval methods, in the prefill stage, total KV caches are stored in the GPU memory (Tang et al., 2024), or offloaded to CPU to reduce memory overhead (Sun et al., 2024; Chen et al., 2024b), and important KV caches are retrieved during the decode stage. Quest (Tang et al., 2024), ShadowKV (Sun et al., 2024) and ArkVale (Chen et al., 2024b) summarize the Key cache chunks into compact representations with different ways (e.g., mean-pooling) and retrieve important KV caches in a chunk-wise manner. Mag-

icPig (Chen et al., 2025b) applies locality-sensitive hashing (LSH) to sample and retrieve significant tokens. TailorKV (Yao et al., 2025) adopts a coarse-grained layer-level hybrid strategy with static quantization and dynamic retrieval. These methods achieve better model performance at the cost of complexity and efficiency. HybridKV uses the basic strategy of chunk-wise retrieval and is orthogonal to the specific optimizations of these methods. The above methods for LLMs focus on text-based KV cache compression while overlooking multimodal contexts (Feng et al., 2025c,a; Wan et al., 2024a; Xiong et al., 2025a,b). They are not directly applicable to multimodal models due to the unique characteristics of cross-modal attention.

As multimodal understanding scenarios progress, KV cache compression methods are further designed for MLLMs and can be divided into three categories: token-level compression, layer-level compression, and head-level compression. For token-level methods, LOOK-M (Wan et al., 2024b) prioritizes the retention of text tokens over visual tokens for modality-aware compression. For layer-level methods, MadaKV (Li et al., 2025) proposes modality preference metric and applies hierarchical compression for attention layers. MEDA (Wan et al., 2025) leverages cross-modal attention entropy for dynamic KV cache allocation across layers. For head-level methods, SparseMM (Wang et al., 2025) observes the visual sparsity of attention heads in MLLMs and allocates asymmetric KV cache budgets to heads based on their visual scores. However, the above methods (Liu et al., 2025a), whether designed for LLMs or MLLMs, treat all the attention heads either in a static manner or in a dynamic manner, leading to suboptimal compression results. In contrast, HybridKV is tailored for optimizing MLLMs and observes the heterogeneous attention patterns across heads. It designs a novel hybrid KV cache compression framework for MLLMs to achieve better model performance with significant efficiency gains.

Visual Token Compression for MLLMs. Existing research proposes various visual token compression methods (Chen et al., 2025a; Han et al., 2026; Liu et al., 2026; Lin et al., 2026) due to the great redundancy of visual tokens. FastV (Chen et al., 2024a) prunes visual tokens after Layer 2 using attention scores. SparseVLM (Zhang et al., 2025c) uses text-to-vision attention for scoring to evict unimportant visual tokens. VisionZip (Yang et al., 2025) reduces visual redundancy in the vision encoders. LightVLM (Hu et al., 2025) proposes pyramid token merging in the prefill stage and KV cache compression in the decode stage. There also exist methods that focus on video inputs because of the high volume of video tokens. Dycoke (Tao et al., 2025) applies token merging across video frames and reduces KV cache dynamically. VidCom^2 (Liu et al., 2025b) dynamically adjusts compression intensity based on frame uniqueness. Differently, our method operates on another perspective that leverages the heterogeneous attention head patterns embedded in MLLMs to address the visual redundancy issues.

C Details of Benchmark

To comprehensively evaluate our model, we utilize a series of established benchmarks for both image and video understanding. For image tasks, we select a multifaceted suited of tasks from MileBench. The selection includes visual difference caption on the CLEVR-Change (CL-CH) (Park et al., 2019) and Spot-the-Diff (STD) (Jhamtani and Berg-Kirkpatrick, 2018) datasets, measured by ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation-Longest Common Subsequence) (Lin, 2004). It also incorporates a diverse set of visual question answering (VQA) tasks that span various domains: document and presentation understanding (DocVQA (Mathew et al., 2021), SlideVQA (Tanaka et al., 2023)), web-based knowledge extrantion (WebQA (Chang et al., 2022), WikiVQA (Yang et al., 2015)), and joint reasoning over text, tables, and images (MMCoQA (Li et al., 2022), MultiModalQA (MM-QA) (Talmor et al., 2021). Performance across all these VQA tasks is measured by accuracy. For video tasks, we employ VATEX (Wang et al., 2019) for video captioning, with performance measured by four metrics: BLEU-4 (4-gram Bilingual Evaluation Understudy) (Papineni et al., 2002), METEOR (Metric for Evaluation of Translation with Explicit ORdering) (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), and CIDEr (Consensus-based Image Description) (Vedantam et al., 2015). For NextQA (Xiao et al., 2021), which focuses on temporal and causal reasoning, we use the metric Wu-Palmer Similarity (WUPS) score (Malinowski and Fritz, 2014) to evaluate the quality of the generated answers. Additionally, we leverage the Video-ChatGPT (Maaz et al., 2024) to evaluate conversational abilities across five dimensions: Correctness of Information (CI), Detail Orientation (DO), Contextual Understanding (CU), Temporal Understanding (TU) and Consistency (CO). These metrics are

Method	WikiVQA	VATEX	Video-ChatGPT
Latency (ms/token)			
FULL CACHE	46.27	51.90	58.94
HYBRIDKV	**33.30**	**37.21**	**38.65**
Memory (GB)			
FULL CACHE	1.32	1.40	1.73
HYBRIDKV	**0.16**	**0.18**	**0.22**

Table 8: Efficiency results on `Qwen2.5-VL-7B`.

Method	WikiVQA	VATEX	Video-ChatGPT
Latency (ms/token)			
FULL CACHE	35.44	46.80	46.80
HYBRIDKV	**28.48**	**35.23**	**33.34**
Memory (GB)			
FULL CACHE	1.42	1.59	1.59
HYBRIDKV	**0.18**	**0.21**	**0.21**

Table 9: Efficiency results on `LLaVA-OneVision-7B`.

accessed using an LLM-generated prediction score ranging from 0 to 5 (GPT Score) with gpt-4o-mini.

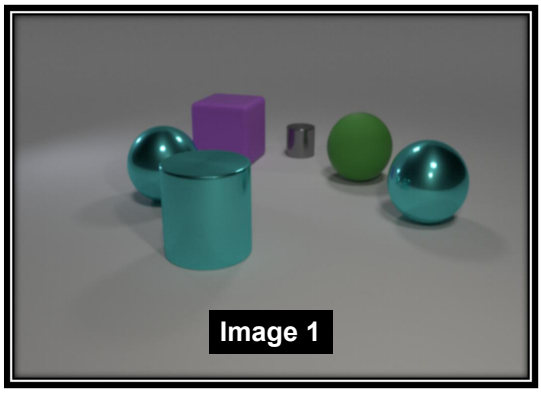
D Additional Experimental Results

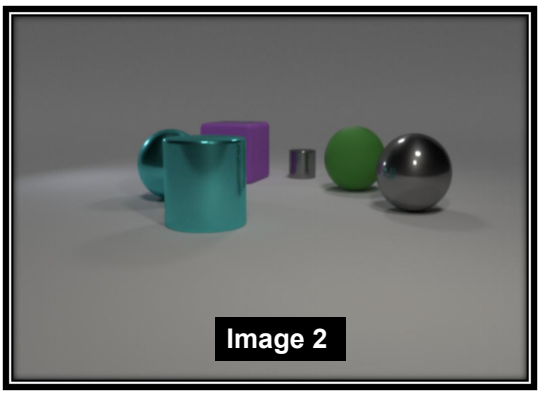
We expand our efficiency evaluation beyond `Qwen2.5-VL-7B`. Specifically, we conduct additional experiments on the `LLaVA-OneVision-7B` model to verify architectural generalizability. Furthermore, to assess performance across diverse modalities and datasets, we extend our evaluation to include WikiVQA (image benchmark) and VATEX (video benchmark). Table 8 and Table 9 present the latency and memory results for `Qwen2.5-VL-7B` and `LLaVA-OneVision-7B`, respectively.

Across three multimodal benchmarks encompassing both image and video tasks, we achieved a $1.39\times$–$1.52\times$ reduction in latency and a $7.8\times$–$8.3\times$ reduction in KV cache GPU memory usage on `Qwen2.5-VL-7B`. Similarly, on `LLaVA-OneVision-7B`, we observed a $1.24\times$–$1.40\times$ latency reduction and a $7.6\times$–$7.9\times$ reduction in memory usage. These results demonstrate the generalizability of HYBRIDKV across different models and tasks.

E Case Study

Two illustrative examples for image and video tasks are presented in Fig. 7 and Fig. 8, respectively. HYBRIDKV achieves superior generation quality by enabling the model to focus on critical visual regions through a head-level hybrid design. This allows it to not only outperform other KV cache compression baselines, but also surpass the model with full cache.





User	*Identify the alterations between these two images.*	
Qwen2.5-VL (Full Cache)	The large **metal sphere** on the **right side** of the image changed to **gray**.	☺
Qwen2.5-VL w/HybridKV	The large **metal sphere** on the **right side** of the image changed to **gray**.	☺
Qwen2.5-VL w/SnapKV	The large metal object has changed to silver.	☹
Qwen2.5-VL w/LOOK-M	The large gray object in front of the green object has changed to a shiny gray object.	☹
Qwen2.5-VL w/MadaKV	The large gray shiny object has been added.	☹
Qwen2.5-VL w/SparseMM	The large shiny object has changed to gray.	☹

Figure 7: Case study of HybridKV on CL-CH using `Qwen2.5-VL-7B`. Other methods generate the *wrong* answer after KV cache compression, while HybridKV still retains the correct answer.

User	*Who is the player being pointed by the red arrow?*	
Qwen2.5-VL (Full Cache)	The video does not provide specific details about who the player being pointed by the red arrow is, as it only shows a general view of the field and players in action..	☹
Qwen2.5-VL w/HybridKV	The red arrow in the video points to a player wearing a **white jersey** with the **number 15** on it. The video highlights his freshman year achievements, including **scoring 8 goals and assisting 6 times**.	☺
Qwen2.5-VL w/SnapKV	The red arrow in the video points to a player wearing a black jersey with the number 22 on it.	☹
Qwen2.5-VL w/LOOK-M	The red arrow in the video points to a player wearing a green jersey with the number 22 on it	☹
Qwen2.5-VL w/MadaKV	The video does not provide specific details about who the player being pointed by the red arrow is, as it only shows a general view of the field and players in action.	☹
Qwen2.5-VL w/SparseMM	The video does not provide specific details about who the player being pointed by the red arrow is.	☹

Figure 8: Case study of HybridKV on Video-ChatGPT using `Qwen2.5-VL-7B`. HybridKV enables the decode tokens to adaptively focus on critical visual regions, highlighted by red font and boxes, while the model with full cache attend to the noisy tokens, leading to degraded accuracy. This demonstrates that retaining a small yet salient subset of tokens during decoding can not only preserve but even enhance the model's capabilities.