

# DIVE: Accurate and Efficient Video Large Language Models for Lifespan Question via Head-wise Visual KV Cache Compression

Sanghwa Hong, Jongjin Kim, U Kang<sup>†</sup>  
Seoul National University  
{hongsw5911, j2kim99, ukang}@snu.ac.kr

## Abstract

Given a Video Large Language Model (VLLM) and a pre-recorded video, how can we minimize the cumulative inference latency over future questions? Addressing this is crucial since VLLMs not only serve a massive volume of questions over the video’s lifespan but also incur substantial latency for each question. Existing methods compress visual tokens generated from a video to accelerate inference for each question. However, under aggressive compression ratios, prior approaches suffer from severe accuracy degradation, still resulting in high inference latency on each question.

In this paper, we propose DIVE, a visual KV cache compression method that preserves accuracy even under aggressive compression ratios. To maintain accuracy, DIVE conserves the temporal context of the input video by evicting KV pairs after the prefill stage, rather than filtering out visual tokens before the VLLM processes the tokens through attention mechanisms. DIVE significantly reduces per-question decoding latency by carefully evicting KV pairs, thereby achieving low cumulative latency over a massive volume of questions. Experiments show that DIVE retains 98.2% of accuracy, using only 5% of the KV cache in Qwen2.5-VL.

## 1 Introduction

Given a video, how can a VLLM efficiently answer a large volume of questions over the video’s lifetime? VLLMs achieve strong performance on question answering with pre-recorded videos (Maaz et al., 2024). However, as their performance increases, the size and complexity of VLLMs also grow (Li et al., 2025; Bai et al., 2025), resulting in significant inference latency for processing each question (Di et al., 2025; Wu et al., 2024b). This

<sup>†</sup>Corresponding author.

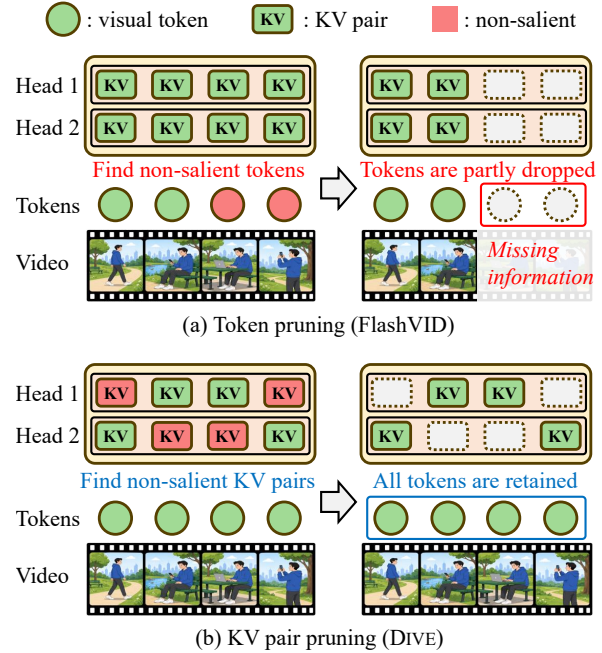


Figure 1: (a) Previous methods remove a subset of visual tokens, discarding their corresponding KV pairs across all heads and losing some temporal segments. (b) DIVE evicts KV pairs in head-wise manner, preserving temporal coverage.

becomes a critical issue in real-world applications where VLLMs need to answer diverse questions from numerous users about a given video (Maaz et al., 2024). To address this challenge, it is essential to compress information so that VLLMs answer each question with low latency.

KV cache compression is a promising approach to reduce VLLM inference latency by pruning the KV cache generated from the video (Qian et al., 2024). Reducing the amount of the KV cache decreases the computation required for each question, thereby reducing per-question latency (Li et al., 2024b). To preserve accuracy after pruning, it is crucial to identify and remove non-essential KV pairs (Yang et al., 2024a). Existing methods identify those by finding non-salient visual tokens from the video and pruning their corresponding KV pairs, as shown in Figure 1(a) (Fan et al., 2026).

However, since this approach completely discards information from certain parts of the video, the accuracy drops significantly as the compression ratio increases (Kim et al., 2025a).

In this paper, we propose DIVE, an accurate KV cache compression method to preserve accuracy under high compression ratios. To prevent the complete loss of information from specific temporal segments, DIVE selects important KV pairs independently for each attention head, as illustrated in Figure 1(b). This allows different heads to retain important information from different timestamps and regions, maintaining accuracy even under high compression ratios. To identify important KV pairs for each head, DIVE first evaluates their importance using attention scores obtained from teacher-forced inference on proxy temporal QAs, and then selects both salient and non-redundant subsets by a Determinantal Point Process (DPP).

The main contributions are as follows:

- **Method.** We propose DIVE, an accurate and efficient visual KV cache compression method that estimates each KV pair’s saliency via teacher-forced proxy QAs.
- **Accuracy.** DIVE achieves state-of-the-art accuracy over previous video compression methods, retaining 98.2% of full-cache accuracy with only 5% of the KV cache.
- **Efficiency.** DIVE achieves  $1.16\times$  faster latency than the best competitor.

The codes are available at <https://github.com/snudatalab/Dive>.

## 2 Preliminaries and Related Works

### 2.1 Problem Definition

**Problem 1** (Lifespan video question answering.). **Given** a pre-recorded video  $V$ , a VLLM  $M$ , and a sequence  $\mathcal{Q} = \{Q_1, Q_2, \dots\}$  of future questions that continually arrive over the lifetime of the video, **minimize** the cumulative inference latency over  $\mathcal{Q}$ :

$$\min \sum_{Q_i \in \mathcal{Q}} \text{Latency}(M, V, Q_i)$$

subject to maintaining accuracy. Note that, unlike conventional single-question acceleration problems, a massive number of questions in  $\mathcal{Q}$  arrive sequentially over the lifetime of the video, and each question must be answered upon arrival without access to future questions.

### 2.2 Video Large Language Models

A VLLM  $M$  ingests a video  $V$  by decomposing it into frames, which a vision encoder and a lightweight projector map into a sequence of visual tokens  $\mathbf{X}^v = (\mathbf{x}_1^v, \dots, \mathbf{x}_N^v)$ , where  $N$  denotes the number of visual tokens per video. These are concatenated with the tokenized question  $\mathbf{X}^q$  and forwarded through  $L$  transformer layers, which autoregressively decode the answer tokens  $\mathbf{X}^a$ .

Modern VLLMs adopt grouped-query attention (GQA) within each transformer layer: every layer maintains  $H$  query heads but only  $G$  KV heads with  $G \leq H$ , where each KV head is shared across  $H/G$  query heads (Ainslie et al., 2023). For layer  $l \in \{1, \dots, L\}$  and KV head  $g \in \{1, \dots, G\}$ , position  $i \in \{1, \dots, N\}$  produces a key  $\mathbf{k}_i^{(l,g)} \in \mathbb{R}^d$  and a value  $\mathbf{v}_i^{(l,g)} \in \mathbb{R}^d$ , which together form the KV pair  $\text{kv}_i^{(l,g)} = (\mathbf{k}_i^{(l,g)}, \mathbf{v}_i^{(l,g)})$ . The collection of KV pairs across  $N$  visual tokens at a given  $(l, g)$  constitutes the per-head visual KV cache  $\mathbf{KV}_N^{(l,g)}$ , and the union of all per-head visual KV caches forms the visual KV cache  $\mathbf{KV}_N$ .

During decoding, for each layer  $l$  and KV head  $g$ , the VLLM maintains cached keys and values over the current input prefix as  $\mathbf{K}^{(l,g)} = [\mathbf{k}_1^{(l,g)}, \dots, \mathbf{k}_N^{(l,g)}, \dots]$  and  $\mathbf{V}^{(l,g)} = [\mathbf{v}_1^{(l,g)}, \dots, \mathbf{v}_N^{(l,g)}, \dots]$ . Let  $g(h) = \lceil hG/H \rceil$  map each query head  $h$  to its associated KV head. At the current decoding position, query head  $h$  produces a query vector  $\mathbf{q}^{(l,h)} \in \mathbb{R}^d$  and the attention output reads:

$$\mathbf{o}^{(l,h)} = (\mathbf{V}^{(l,g(h))})^\top \text{softmax}\left(\frac{\mathbf{K}^{(l,g(h))} \mathbf{q}^{(l,h)}}{\sqrt{d}}\right).$$

### 2.3 Instruction Tuning

Large Language Models (LLMs) pretrained with a next-token prediction objective assign high likelihood to fluent continuations, but does not align with user intent (Li et al., 2022; Ouyang et al., 2022; Kim et al., 2026b,a, 2025b; Jeon et al., 2023; Cho and Kang, 2022; Piao et al., 2022). Instruction tuning addresses this by fine-tuning on instruction-response pairs, aligning generation with user intent (Wei et al., 2022; Yun et al., 2026; Park et al., 2025a,b, 2024; Jeon et al., 2025). During this supervised stage, teacher forcing is employed: ground-truth response tokens are provided as inputs, and the model is trained to predict each subsequent response token given the instruction. Although instruction tuning is conducted on a finite set of

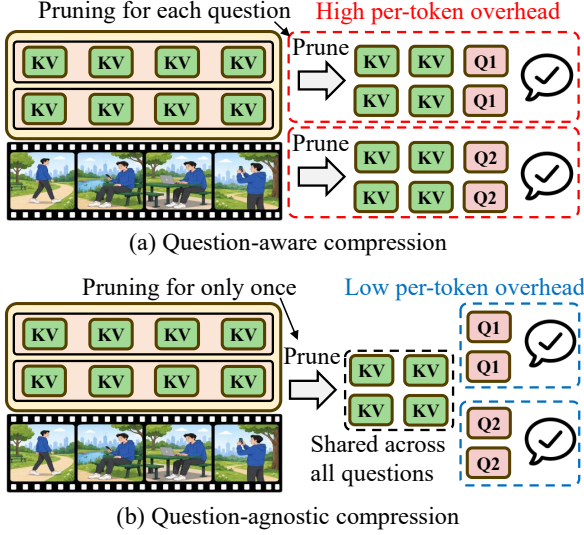


Figure 2: (a) Question-aware compression prunes the KV cache for each question separately, resulting in significant per-question overhead that accumulates over time. (b) Question-agnostic compression prunes the KV cache only once, saving per-question overhead and reducing cumulative latency.

training instructions, its objective is not merely to memorize these instances, but to induce a general instruction-following capability that transfers to unseen user questions (Longpre et al., 2023). Generalization strengthens as the number of QA pairs and instruction-formatted tasks increase (Chung et al., 2024). Self-Instruct (Wang et al., 2023) further shows that synthetically generated instruction-response pairs are effective surrogates for ground-truth QA pairs.

## 2.4 Question-aware Compression

Question-aware compression, described in Figure 2(a), retains only the most relevant KV cache for each question and pruning the rest to accelerate inference (Huang et al., 2025). To determine which KV cache to prune, it evaluates the importance of each KV pair respect to the question, using attention scores or similar metrics (Zhang et al., 2023; Li et al., 2024b). This approach is advantageous to maintain accuracy under high compression ratios, as it selects the most relevant KV cache for each question (Tang et al., 2024; Xiao et al., 2024). However, it has limitations to reduce cumulative latency in the lifespan QA scenario, since it needs to select a new KV cache for each incoming question, incurring per-question compression overhead. Thus, to reduce cumulative latency for a large number of questions over time (Yang et al., 2024b,c; Di et al., 2025), we focus on question-agnostic compression instead of question-aware compression.

## 2.5 Question-agnostic Compression

Question-agnostic compression in Figure 2(b) preserves only a subset of important KV pairs before receiving any questions, and then answers subsequent questions using the compressed KV cache (Shen et al., 2025; Fan et al., 2026). Unlike question-aware methods, this approach avoids the overhead of additional KV cache selection for every question. Thus, it is particularly suitable for lifespan QA scenarios, where numerous questions are asked over time and reducing the cumulative latency is crucial.

To maintain accuracy, it is important to correctly identify the core KV cache that captures the essential information of the video without access to the questions. Existing methods first identify non-salient visual tokens and subsequently remove their corresponding KV pairs (Chen et al., 2024; Yang et al., 2025). For example, prior methods treat redundant visual tokens as non-essential and prune the KV pairs associated with those tokens (Shen et al., 2025; Fan et al., 2026). However, they operate at the token level, discarding all KV pairs tied to the pruned visual tokens. We address this limitation by proposing a token-independent KV cache compression method that selectively retains informative KV pairs within the visual KV cache, as described in Section 3.1.

## 3 Proposed Method

### 3.1 Overview

We propose DIVE (Head-wise visual KV cache compression), an accurate KV cache compression method that prunes KV cache in a head-wise manner to prevent the complete loss of information from specific temporal segments. To sustain the accuracy from the compression, we address the following three challenges:

**C1. Loss of attention heads.** Attention-based importance scores tend to concentrate on a small subset of heads. Hence, naively pruning KV pairs across all heads leads to entirely discarding certain heads, reducing the effectiveness of multi-head attention. However, the compressed KV cache should retain information across diverse heads, since each incoming question relies on a distinct subset of attention heads. How can we maintain the effectiveness of multi-head attention?

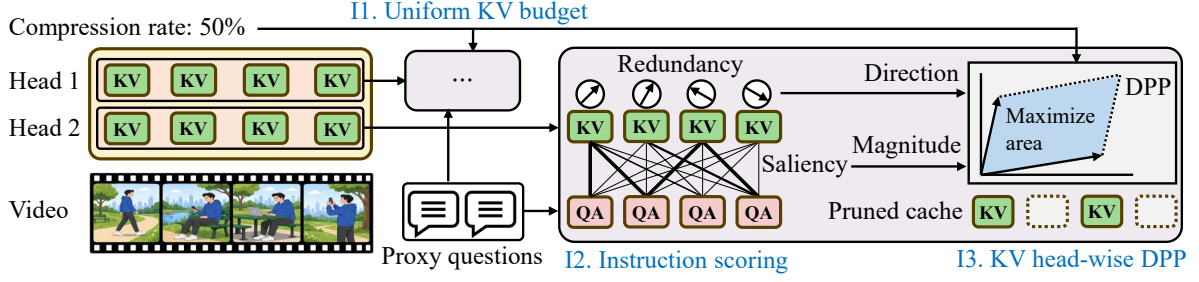


Figure 3: Overview of DIVE.

- C2. Finding salient KV pairs.** Existing question-agnostic KV cache compression methods rely on question-free objectives such as key/value norms to find salient KV pairs. However, they struggle to identify KV pairs that are genuinely important for future questions since they do not evaluate the usefulness in the context of answering actual questions. How can we precisely estimate saliency for unseen questions?
- C3. Redundancy within KV pairs.** Existing KV cache methods retain KV pairs whose key representation deviates the most from a mean key representation. However, they neglect pairwise similarity among the retained pairs, resulting in the selected subset exhibiting high redundancy. How can we select a KV subset that is both salient and mutually non-redundant?

We address the aforementioned challenges by the following ideas:

- I1. Uniform KV budget (Section 3.2)** DIVE assigns the same KV budget across every layer-KV head pair, preventing the retained KV cache from being concentrated only in a few heads.
- I2. Instruction scoring (Section 3.3)** DIVE teacher-forces proxy QAs when measuring the saliency of each visual KV pair. The proxy QAs provide question-answering contexts that allow DIVE to estimate which KV pairs are important for answering unseen future questions.
- I3. KV head-wise DPP (Section 3.4)** DIVE penalizes pairwise redundancy among KV pairs, thereby producing a sufficiently informative KV subset for serving diverse future questions.

Figure 3 provides an overview of DIVE. Given the prefilled visual KV pairs from a video, the **uniform KV budget** constrains each layer-KV

head pair to retain an identical number of KV pairs. The **instruction scoring** takes a video and a set of proxy temporal QAs as input, and produces saliency scores for visual KV pairs at each query head. For each KV head, the **KV head-wise DPP** receives the estimated saliency scores of visual KV pairs across query heads and their key embeddings, and produces the indices of KV pairs to retain.

### 3.2 Uniform KV Budget

How can we prevent entire attention heads from being lost when we prune the KV cache at the level of individual KV pairs, instead of tokens? A naive approach to prune KV pairs is to rank them based on attention scores and evicting low-scoring pairs across attention heads. However, this approach starves certain attention heads because attention scores tend to concentrate on a small subset of heads (Fu et al., 2025). Considering that different questions rely on distinct subsets of attention heads, such head loss degrades accuracy when serving varied future questions (Michel et al., 2019). To address this, DIVE assigns an identical KV budget to every layer-KV head pair. This prevents the compressed KV cache from being concentrated in only a few heads, thereby maintaining accuracy across future questions that rely on different heads (Voita et al., 2019).

Given the prefilled visual KV pairs generated from a video, the **uniform KV budget** constrains per-head visual KV cache  $\mathbf{KV}_N^{(l,g)}$  at each layer  $l$  and KV head  $g$  to retain  $B$  visual KV pairs out of its prefilled  $N$  visual KV pairs. We set  $B = \lfloor \rho N \rfloor$ , where  $\rho \in (0, 1]$  is the target retention ratio. See Appendix A for a justification of the uniform KV budget.

### 3.3 Instruction Scoring

How can we precisely estimate saliency of KV pairs for unknown future questions? Existing KV cache compression methods estimate KV-pair saliency using question-free proxy objectives, such

as key/value norms or context reconstruction (Kim et al., 2025a). However, saliency estimated from question-free signals does not reflect which KV pairs are genuinely useful for unseen future questions, as the scoring is not conditioned on the actual question-answering context (Wang et al., 2025). To address this, DIVE generates a set of proxy temporal QAs from the video and teacher-forces them when scoring KV-pair saliency. The proxy QAs introduce question-answering contexts, allowing DIVE to infer which KV pairs are likely to be useful for unseen future questions.

Given a video and synthetically generated proxy temporal QAs, **instruction scoring** produces the saliency of visual KV pairs. To estimate the saliency of each visual KV pair, DIVE measures how strongly the KV pair is referenced when answering proxy QAs. Specifically, DIVE uses the maximum attention score it receives from teacher-forced QA-derived query embeddings as its saliency score.

The procedure below explains how DIVE estimates the saliency score  $s_i^{(l,g)}$  for each visual KV pair  $i \in \{1, \dots, N\}$  at layer  $l$  and KV head  $g$ , using the proxy QA set  $\mathcal{Q}^p = \{(Q_t, A_t)\}_{t=1}^T$ .  $Q_t$  and  $A_t$  denote the question and answer of the  $t$ -th proxy QA, respectively. To estimate the score, DIVE computes per-proxy-QA saliency score  $s_i^{(l,g,t)}$  for each  $t$ -th proxy QA, then takes the maximum over all proxy QAs:

$$s_i^{(l,g)} = \max_{t \in \{1, \dots, T\}} s_i^{(l,g,t)}.$$

Using multiple proxy QAs enables DIVE to accurately estimate saliency.

For the  $t$ -th proxy QA  $(Q_t, A_t)$ , DIVE computes the per-proxy-QA saliency score  $s_i^{(l,g,t)}$  as follows. DIVE teacher-forces  $(Q_t, A_t)$  and performs a single forward pass. Since the  $N$  visual tokens occupy positions  $1, \dots, N$ , the appended token positions of the  $t$ -th proxy QA are defined as

$$\mathcal{R}(t) = \{N + 1, \dots, N + |Q_t| + |A_t|\},$$

where  $|Q_t|$  and  $|A_t|$  are the token counts of the question  $Q_t$  and answer  $A_t$ , respectively. After the teacher-forced forward pass, DIVE computes the pre-softmax attention score  $\alpha_{r,i}^{(l,h,t)}$ , which measures how strongly the appended QA token at position  $r \in \mathcal{R}(t)$  attends to the visual KV pair at

position  $i$ :

$$\alpha_{r,i}^{(l,h,t)} = \frac{\left(\mathbf{k}_i^{(l,g(h))}\right)^\top \mathbf{q}_r^{(l,h,t)}}{\sqrt{d}}.$$

Here,  $\mathbf{q}_r^{(l,h,t)}$  denotes the QA-token query embedding of the appended token at position  $r$ , produced at layer  $l$ , query head  $h$ , and the  $t$ -th proxy QA.  $\mathbf{k}_i^{(l,g(h))}$  denotes the visual-key embedding of the visual KV pair at position  $i$ , produced at layer  $l$  and KV head  $g(h)$ , where  $g(h)$  maps query head  $h$  to its associated KV head as defined in Section 2.2. For the query-head set  $\mathcal{H}(g) \subseteq \{1, \dots, H\}$  grouped to KV head  $g$  and the appended positions  $\mathcal{R}(t)$ , the per-proxy-QA saliency score is defined as

$$s_i^{(l,g,t)} = \max_{(h,r) \in \mathcal{H}(g) \times \mathcal{R}(t)} \alpha_{r,i}^{(l,h,t)}.$$

DIVE uses a number of proxy QAs spanning diverse QA tasks when scoring KV-pair saliency to distinguish KV pairs that are useful across diverse questions. This design is motivated by instruction tuning: LLMs’ ability to answer unseen questions improves as the number of QA pairs and diversity of instruction-formatted tasks increase (Section 2.3).

To synthesize proxy temporal QAs, DIVE first extracts temporal events from the video using an off-the-shelf video captioning model and then generates QA pairs with predefined templates. Although the QAs are synthetic, (Wang et al., 2023) shows that synthetically generated QA pairs serve as effective surrogates for ground-truth QA. See Appendix B for details on QA generation.

### 3.4 KV head-wise DPP

How can we select a KV subset which is both salient and mutually non-redundant? Similar KV pairs reduce the diversity of information retained in the KV cache, which degrades accuracy under the same compression ratio. Existing methods prioritize KV pairs whose keys deviate more from the average key representation to reduce redundancy within the selected KV pairs. However, this approach has limitations in fully removing redundancy, since they do not account for pairwise similarity among the KV pairs within the selected subset. DIVE instead penalizes pairwise redundancy among retained KV pairs, producing an informative per-head cache.

To address this, for each layer  $l$  and KV head  $g$ , the KV head-wise DPP takes the KV-head-specific saliency scores  $\{s_1^{(l,g)}, \dots, s_N^{(l,g)}\}$  produced by Section 3.3 and the key embeddings  $\{\mathbf{k}_1^{(l,g)}, \dots, \mathbf{k}_N^{(l,g)}\}$  of the KV pairs within KV head, and yields the compressed per-head visual KV cache  $\mathbf{KV}_B^{(l,g)}$  that retains  $B$  KV pairs out of  $N$  visual KV pairs. Specifically, DIVE selects a size- $B$  subset  $\mathbf{KV}_B^{(l,g)}$  whose normalized key representations span a large volume, so that KV pairs with similar key representations are less likely to be selected together.

This selection is formulated as the following DPP-style (Kulesza and Taskar, 2012) optimization problem: given a matrix  $\mathbf{L}^{(l,g)} \in \mathbb{R}^{N \times N}$  (see Eq. (1)), DIVE finds a principal submatrix  $\mathbf{L}_S^{(l,g)} \in \mathbb{R}^{B \times B}$  such that  $\mathcal{S}$  is a size- $B$  subset of  $\{1, \dots, N\}$  and  $\mathbf{L}_S^{(l,g)}$  has the maximum determinant:

$$\mathcal{S}^{(l,g)*} = \arg \max_{\mathcal{S} \subseteq \{1, \dots, N\}, |\mathcal{S}|=B} \det(\mathbf{L}_S^{(l,g)}).$$

Maximizing this determinant selects the position set  $\mathcal{S}^{(l,g)*}$  whose corresponding KV pairs are not only salient but also informative, encouraging their key representations to be closer to mutually orthogonal than redundant.

The quality-weighted pairwise similarity  $\mathbf{L}_{ij}^{(l,g)}$ , the  $(i, j)$ -th element of  $\mathbf{L}^{(l,g)}$ , is defined as

$$\mathbf{L}_{ij}^{(l,g)} = q_i^{(l,g)} \phi_{ij}^{(l,g)} q_j^{(l,g)}, \quad (1)$$

where  $\phi_{ij}^{(l,g)}$  measures the pairwise key similarity between KV pairs  $i$  and  $j$  while  $q_i^{(l,g)}$  encodes the saliency-based quality of KV pair  $i$  (see Eq. (2) and Eq. (3)). The  $\mathbf{L}_{ij}^{(l,g)}$  acts as a quality-weighted overlap term: large qualities  $q_i^{(l,g)}$  and  $q_j^{(l,g)}$  increase the determinant, while large  $\phi_{ij}^{(l,g)}$  reduces the determinant. For example, when selecting two positions  $\mathcal{S} = \{1, 2\}$ , we have  $\det(\mathbf{L}_S^{(l,g)}) = (q_1^{(l,g)})^2 (q_2^{(l,g)})^2 (1 - (\phi_{12}^{(l,g)})^2)$ , showing that the determinant is proportional to the squared qualities and becomes larger as the selected key embeddings become more orthogonal.

DIVE computes  $\phi_{ij}^{(l,g)}$  using cosine similarity:

$$\phi_{ij}^{(l,g)} = \frac{\langle \mathbf{k}_i^{(l,g)}, \mathbf{k}_j^{(l,g)} \rangle}{\|\mathbf{k}_i^{(l,g)}\| \|\mathbf{k}_j^{(l,g)}\|}. \quad (2)$$

With hyperparameter  $\beta > 0$ , DIVE defines the quality term as

$$q_i^{(l,g)} = \exp(\beta \tilde{s}_i^{(l,g)}), \quad (3)$$

where  $\tilde{s}_i^{(l,g)}$  is the min-max normalized saliency:

$$\tilde{s}_i^{(l,g)} = \frac{s_i^{(l,g)} - \min_j s_j^{(l,g)}}{\max_j s_j^{(l,g)} - \min_j s_j^{(l,g)}}.$$

See Appendix C for details of solver.

## 4 Experiments

Experiments are structured to answer the following research questions:

- RQ1 Accuracy (Section 4.2).** Does DIVE preserve accuracy under high compression ratios?
- RQ2 Per-token latency (Section 4.3).** How much faster does DIVE decode tokens than existing methods?
- RQ3 Cumulative latency (Section 4.4).** Is DIVE effective for large-scale video question serving?
- RQ4 Ablation study (Section 4.5).** Does each module of DIVE independently improve accuracy?
- RQ5 Instruction scoring (Section 4.6).** Does instruction scoring become more effective as the number or types of proxy QA change?

### 4.1 Experimental Setup

**Benchmarks.** We evaluate DIVE on five widely-used video understanding benchmarks: VMME (VideoMME) (Fu et al., 2024), ES (EgoSchema) (Mangalam et al., 2023), LVB (LongVideoBench) (Wu et al., 2024a), MVB (MVBench) (Li et al., 2024a), and MLVU (Zhou et al., 2024). Details are in Appendix D.

**Compared Baselines.** We compare DIVE with five methods: FastV (Chen et al., 2024), VisionZip (Yang et al., 2025), PruneVID (Huang et al., 2025), FastVID (Shen et al., 2025), and FlashVID (Fan et al., 2026). Implementation details are in Appendix E.

**Implementation Details.** We evaluate DIVE on LLaVA-OneVision (Li et al., 2025) and Qwen2.5-VL (Bai et al., 2025). LLaVA-OneVision uniformly samples 32 frames, producing  $32 \times 196$  visual tokens. We sweep the retention ratio over  $\{25\%, 20\%, 15\%, 10\%, 5\%\}$ . The evaluation uses

Table 1: Accuracy of DIVE and competitors. “Ret.” denotes the retained KV-cache percentage, “Rel. Acc” denotes accuracy relative to the vanilla VLLM in percent, and bold denotes the best.

| Model              | Method                 | Ret. | VMME        |             |             |             | ES          |             | LVB         | MVB         | Avg.        |              |
|--------------------|------------------------|------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|--------------|
|                    |                        |      | Short       | Medium      | Long        | Overall     | Subset      | Total       | val         | Overall     | Score       | Rel. Acc (%) |
| LLaVA<br>OneVision | Vanilla                | 100% | <b>69.9</b> | <b>56.7</b> | <b>48.9</b> | <b>58.5</b> | <b>62.2</b> | <b>60.3</b> | <b>56.6</b> | <b>58.3</b> | <b>58.4</b> | <b>100.0</b> |
|                    | FastV                  | 25%  | 68.1        | 54.7        | 46.8        | 56.5        | 60.4        | 57.8        | 55.4        | 56.4        | 56.5        | 96.7         |
|                    | VisionZip              |      | 68.8        | <b>57.3</b> | 48.2        | 58.1        | 63.0        | 60.4        | <b>56.4</b> | 57.8        | 58.2        | 99.7         |
|                    | PruneVID               |      | 67.3        | 54.8        | 47.2        | 56.4        | 61.0        | 58.1        | 55.4        | 56.8        | 56.7        | 97.1         |
|                    | FastVID                |      | 69.9        | 56.3        | 47.4        | 57.9        | 61.2        | 59.5        | 55.9        | 58.1        | 57.8        | 99.0         |
|                    | FlashVID               |      | <b>70.2</b> | 57.1        | 48.6        | <b>58.6</b> | <b>63.4</b> | 60.4        | 56.3        | 58.0        | 58.3        | 99.9         |
|                    | <b>DIVE (proposed)</b> |      | 70.1        | 57.0        | <b>48.7</b> | <b>58.6</b> | 63.2        | <b>61.2</b> | 56.2        | <b>58.6</b> | <b>58.6</b> | <b>100.4</b> |
|                    | FastV                  | 20%  | 66.3        | 53.9        | 46.9        | 55.7        | 60.6        | 57.6        | 56.0        | 56.0        | 56.3        | 96.4         |
|                    | VisionZip              |      | 68.6        | <b>57.0</b> | 48.3        | 58.0        | 62.0        | 60.0        | 55.4        | 57.6        | 57.7        | 98.8         |
|                    | PruneVID               |      | 67.2        | 53.9        | 48.2        | 56.4        | <b>63.2</b> | 60.2        | 55.2        | 56.2        | 57.0        | 97.6         |
|                    | FastVID                |      | 69.9        | 56.3        | 47.4        | 57.9        | 61.2        | 59.5        | 55.9        | 58.1        | 57.9        | 99.1         |
|                    | FlashVID               |      | 70.1        | 55.6        | 48.4        | 58.0        | 62.8        | 60.0        | <b>56.1</b> | 57.5        | 57.9        | 99.2         |
|                    | <b>DIVE (proposed)</b> |      | <b>70.2</b> | 55.5        | <b>48.6</b> | <b>58.1</b> | 62.9        | <b>60.8</b> | 55.8        | <b>58.2</b> | <b>58.2</b> | <b>99.7</b>  |
|                    | FastV                  | 15%  | 64.6        | 54.0        | 45.3        | 54.6        | 59.8        | 56.6        | 54.8        | 55.0        | 55.2        | 94.5         |
|                    | VisionZip              |      | 63.8        | 54.6        | <b>48.3</b> | 55.6        | <b>62.8</b> | 60.0        | 54.1        | 53.5        | 55.8        | 95.5         |
|                    | PruneVID               |      | 67.2        | 54.8        | 46.7        | 56.1        | 61.6        | 57.7        | 54.5        | 55.1        | 55.7        | 95.4         |
|                    | FastVID                |      | <b>69.7</b> | <b>55.8</b> | 47.7        | <b>57.7</b> | 58.8        | 58.9        | <b>56.7</b> | 57.8        | <b>57.9</b> | 99.1         |
|                    | FlashVID               |      | 69.6        | 55.3        | 48.1        | <b>57.7</b> | 62.2        | 59.8        | 56.5        | 57.2        | 57.8        | 99.0         |
|                    | <b>DIVE (proposed)</b> |      | 69.5        | 55.5        | 48.0        | <b>57.7</b> | 62.5        | <b>60.5</b> | 55.6        | <b>58.0</b> | <b>57.9</b> | <b>99.2</b>  |
|                    | FastV                  | 10%  | 60.9        | 52.2        | 44.9        | 52.7        | 59.0        | 56.0        | 52.4        | 53.4        | 53.6        | 91.8         |
|                    | VisionZip              |      | 60.3        | 53.0        | 46.7        | 53.3        | 61.6        | 58.5        | 49.4        | 54.8        | 54.0        | 92.5         |
|                    | PruneVID               |      | 65.9        | 52.8        | 45.6        | 54.9        | 60.0        | 57.2        | 54.0        | 53.7        | 54.9        | 94.0         |
|                    | FastVID                |      | <b>68.1</b> | <b>55.7</b> | 47.8        | <b>57.2</b> | 58.8        | 58.7        | 55.7        | 57.0        | 57.1        | 97.8         |
|                    | FlashVID               |      | 67.3        | 55.3        | <b>47.9</b> | 56.8        | 61.2        | 59.0        | <b>55.9</b> | 56.7        | 57.1        | 97.8         |
|                    | <b>DIVE (proposed)</b> |      | 67.1        | 55.4        | 47.7        | 56.7        | <b>62.0</b> | <b>60.2</b> | 55.2        | <b>57.7</b> | <b>57.5</b> | <b>98.4</b>  |
|                    | FlashVID               | 5%   | <b>66.9</b> | 53.5        | 45.6        | 55.3        | 60.1        | 57.4        | <b>55.3</b> | 53.1        | 55.3        | 94.7         |
|                    | <b>DIVE (proposed)</b> |      | 66.8        | <b>54.6</b> | <b>46.8</b> | <b>56.1</b> | <b>61.6</b> | <b>59.7</b> | 55.1        | <b>56.9</b> | <b>56.9</b> | <b>97.5</b>  |
| Qwen2.5-VL         | Vanilla                | 100% | <b>72.6</b> | <b>61.4</b> | <b>49.9</b> | <b>61.3</b> | <b>60.2</b> | <b>58.3</b> | <b>58.9</b> | <b>68.0</b> | <b>61.6</b> | <b>100.0</b> |
|                    | FastV                  | 20%  | 69.4        | 57.0        | <b>51.2</b> | 59.2        | <b>60.2</b> | 57.1        | 54.2        | 66.8        | 59.3        | 96.3         |
|                    | VisionZip              |      | 69.6        | 57.2        | 50.2        | 59.0        | 58.6        | 56.6        | 56.3        | 66.4        | 59.6        | 96.8         |
|                    | FastVID                |      | 69.9        | 56.3        | 49.7        | 58.6        | 57.2        | 56.4        | 57.8        | 64.7        | 59.4        | 96.4         |
|                    | FlashVID               |      | 70.4        | 58.6        | 49.7        | 59.6        | 59.2        | 56.8        | <b>58.1</b> | 66.5        | 60.2        | 97.7         |
|                    | <b>DIVE (proposed)</b> |      | <b>70.5</b> | <b>60.2</b> | 50.6        | <b>60.4</b> | 59.9        | <b>57.8</b> | 57.4        | <b>68.9</b> | <b>61.1</b> | <b>99.2</b>  |
|                    | FastV                  | 10%  | 63.7        | 54.7        | 49.3        | 55.9        | <b>58.6</b> | 54.9        | 51.1        | 63.6        | 56.4        | 91.6         |
|                    | VisionZip              |      | 67.0        | 54.7        | 47.6        | 56.4        | 55.8        | 55.5        | 54.5        | 64.3        | 57.7        | 93.7         |
|                    | FastVID                |      | 66.3        | 53.6        | 49.0        | 56.3        | 56.0        | 55.6        | 55.4        | 62.3        | 57.4        | 93.2         |
|                    | FlashVID               |      | 68.1        | 54.7        | 49.0        | 57.3        | 57.4        | 55.9        | <b>57.1</b> | 65.5        | 58.9        | 95.6         |
|                    | <b>DIVE (proposed)</b> |      | <b>70.4</b> | <b>59.9</b> | <b>49.6</b> | <b>60.0</b> | 59.5        | <b>58.0</b> | 56.3        | <b>68.7</b> | <b>60.8</b> | <b>98.7</b>  |
|                    | FlashVID               | 5%   | 66.1        | 53.8        | 48.7        | 56.2        | 56.9        | 55.8        | <b>57.0</b> | 64.9        | 58.4        | 94.9         |
|                    | <b>DIVE (proposed)</b> |      | <b>70.2</b> | <b>59.6</b> | <b>49.9</b> | <b>59.9</b> | <b>59.9</b> | <b>57.8</b> | 55.7        | <b>68.4</b> | <b>60.5</b> | <b>98.2</b>  |

LMMs-Eval on A100 GPUs (Zhang et al., 2025a). Seconds Per Token (SPT) denotes the per-token decoding latency. We measure SPT after pre-fill. Time-To-First-Token (TTFT) denotes the time taken to generate the first token. See Appendix F for TTFT and SPT analysis.  $n$  denotes the number of questions. With 256 generated tokens per answer, we compute Cumulative Latency (CL) as  $TTFT + SPT \times 256 \times n$  and Per-question La-

tency (PL) as  $(TTFT + SPT \times 256 \times n)/n$ . See Appendices G and H for the break-even and hyperparameter sensitivity analyses, respectively.

## 4.2 Accuracy (RQ1)

We evaluate whether DIVE preserves accuracy under aggressive KV cache compression ratios. Table 1 shows that DIVE achieves the best average performance even under low retention ratios. No-

Table 2: Latency comparison analysis for Qwen2.5-VL on 16-frame.

| Metric          | Vanilla | FlashVID | DIVE   |
|-----------------|---------|----------|--------|
| Retention ratio | 100%    | 20%      | 5%     |
| Rel. Acc (%)    | 100.0   | 97.7     | 98.2   |
| TTFT (s)        | 0.438   | 0.319    | 27.6   |
| SPT (s/token)   | 0.0351  | 0.0349   | 0.0294 |

Table 3: CL and PL for serving multiple questions. We use setting reported in Table 2.

| $n$  | Vanilla |      | FlashVID |      | DIVE  |       |
|------|---------|------|----------|------|-------|-------|
|      | CL      | PL   | CL       | PL   | CL    | PL    |
| 10   | 90      | 8.99 | 89       | 8.94 | 103   | 10.29 |
| 20   | 180     | 8.99 | 179      | 8.94 | 178   | 8.91  |
| 160  | 1438    | 8.99 | 1430     | 8.94 | 1232  | 7.70  |
| 2560 | 23014   | 8.99 | 22861    | 8.93 | 19302 | 7.54  |
| 5120 | 46007   | 8.99 | 45744    | 8.93 | 38564 | 7.53  |

tably, on Qwen2.5-VL, DIVE retains 98.2% of the full-cache accuracy while using only 5% of the full visual KV cache.

### 4.3 Per-token latency (RQ2)

We evaluate decoding efficiency of DIVE while preserving accuracy. Table 2 shows that DIVE achieves  $1.16\times$  lower SPT than FlashVID while maintaining accuracy. FlashVID suffers from accuracy degradation under aggressive compression, resulting in high SPT than DIVE. In contrast, DIVE preserves accuracy even at a low retention ratio, enabling lower SPT.

### 4.4 Cumulative Latency (RQ3)

We evaluate whether DIVE efficiently serves a massive volume of questions. Table 3 shows that as the number of served questions increases, DIVE increasingly reduces PL while achieving lower CL than FlashVID. Although DIVE has higher CL at  $n = 10$  due to its one-time compression cost, this cost is amortized as more questions are served, yielding lower CL than FlashVID with an increasingly larger gap. At  $n = 20$ , DIVE already achieves lower CL than FlashVID, showing that its one-time compression cost is amortized early. At  $n = 5120$ , Vanilla and FlashVID show almost identical PLs, whereas DIVE reduces PL by 15.7% over FlashVID. These results highlight the limitation of serving lifespan questions merely from the conventional single-question acceleration perspective. In long-term serving, where both accuracy and efficiency matter across a massive number of questions, it is

Table 4: Ablation study at 5% retention on VMME (overall), ES (total), LVB (val), MVB (overall). **I** denotes saliency-greedy selection with instruction scoring; **I+U** adds the uniform per-head KV budget; **I+U+D** adds KV head-wise DPP.

| Model              | Method | VMME | ES   | LVB  | MVB  |
|--------------------|--------|------|------|------|------|
| LLaVA<br>OneVision | I      | 55.2 | 58.8 | 54.1 | 55.9 |
|                    | I+U    | 55.9 | 59.5 | 54.8 | 56.6 |
|                    | I+U+D  | 56.1 | 59.7 | 55.1 | 56.9 |
| Qwen2.5-VL         | I      | 59.1 | 56.7 | 54.9 | 67.6 |
|                    | I+U    | 59.6 | 57.4 | 55.1 | 68.1 |
|                    | I+U+D  | 59.9 | 57.8 | 55.7 | 68.4 |

Table 5: Instruction scoring results on MLVU with LLaVA-OneVision using 16 frames. Rows denote the number of proxy QAs used for scoring, and columns denote the number of QA types used.

| Number of<br>proxy QAs | Number of QA Types |      |      |      |
|------------------------|--------------------|------|------|------|
|                        | 1                  | 2    | 4    | 8    |
| 1                      | 56.1               | –    | –    | –    |
| 2                      | 57.2               | 57.6 | –    | –    |
| 4                      | 58.4               | 58.5 | 58.7 | –    |
| 8                      | 58.8               | 58.9 | 59.0 | 59.0 |
| 16                     | 58.8               | 59.0 | 59.1 | 59.1 |

more beneficial to trade a one-time compression cost for higher accuracy and lower per-token latency.

### 4.5 Ablation Study (RQ4)

We evaluate whether each module proposed in DIVE is independently effective. Table 4 shows the performance changes of DIVE as modules are incrementally added. The results show that each component of DIVE independently and consistently improves performance.

### 4.6 Instruction Scoring (RQ5)

We evaluate how the number and diversity of proxy QAs affect instruction scoring. Table 5 shows that accuracy generally improves as more proxy QAs and more QA types are used for scoring. This trend supports our instruction-scoring perspective: exposing the model to diverse proxy QA contexts enables DIVE to accurately estimate KV-pair saliency.

## 5 Conclusion

We introduce video lifespan question answering, a problem setting whose goal is to minimize the sum of inference latency over future questions about a pre-recorded video. To address this problem, we propose DIVE, an accurate visual KV cache compression method that evicts KV pairs head-wisely. This enables high KV cache compression ratios, achieving low per-question latency in the lifespan serving scenario. This advantage arises from DIVE’s trade-off of a one-time compression cost for improved accuracy and lower per-token latency, making DIVE effective for massive question-serving scenarios where cumulative latency is the primary objective. Comprehensive experiments show that DIVE maintains high accuracy even at extreme compression ratios, resulting in substantial reductions in cumulative inference latency.

## Acknowledgments

This work was supported by Mobile eXperience (MX) Business, Samsung Electronics Co., Ltd. We used AI assistants solely for polishing the language of the manuscript. U Kang is the corresponding author.

## Limitations

First, we assume that the video receives a large number of future questions, which amortizes DIVE’s one-time compression initialization cost. However, DIVE becomes less beneficial for videos that receive only a small number of questions. Second, the uniform KV budget is motivated by the assumption that diverse future questions balance per-head demand over the video lifetime. However, if the question stream is highly skewed toward a narrow set of question types or visual events, the optimal budget allocation deviates from a uniform one. Third, DIVE estimates KV-pair saliency using proxy temporal QAs generated from dense video captions. However, errors or omissions in dense captions propagate to the saliency scores.

## Ethics Statement

We use publicly available benchmarks, including VideoMME, EgoSchema, LongVideoBench, MVBench, and MLVU, following their respective licenses and terms of use. Experiments rely on open-source pretrained VLLMs, LLaVA-OneVision and Qwen2.5-VL, under their official

licenses. DIVE modifies only the KV-cache management of these models at inference time and introduces no additional training data or personal information. We do not collect, use, or expose any private or user-generated data. We do not observe additional ethical or societal risks beyond those already documented for the base models and benchmarks.

## References

- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. [GQA: Training generalized multi-query transformer models from multi-head checkpoints](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.
- Yash Akhauri, Ahmed F. AbouElhamayed, Jordan Dotzel, Zhiru Zhang, Alexander M. Rush, Safeen Huda, and Mohamed S. Abdelfattah. 2024. [ShadowLLM: Predictor-based contextual sparsity for large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19154–19167, Miami, Florida, USA. Association for Computational Linguistics.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao 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, and 8 others. 2025. [Qwen2.5-vl technical report](#). *arXiv preprint arXiv:2502.13923*.
- Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. 2024. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
- Jen-Hao Cheng, Vivian Wang, Huayu Wang, Huapeng Zhou, Yi-Hao Peng, Hou-I Liu, Hsiang-Wei Huang, Kuang-Ming Chen, Cheng-Yen Yang, Wenhao Chai, Yi-Ling Chen, Vibhav Vineet, Qin Cai, and Jenq-Neng Hwang. 2025. [TEMPURA: Temporal event masked prediction and understanding for reasoning in action](#). *arXiv preprint arXiv:2505.01583*.
- Ikhyun Cho and U Kang. 2022. [Pea-KD: Parameter-efficient and accurate knowledge distillation on BERT](#). *PLOS ONE*, 17(2):e0263592.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16

- others. 2024. [Scaling instruction-finetuned language models](#). *Journal of Machine Learning Research*, 25(70):1–53.
- Douglas E. Crabbtree and Emilie V. Haynsworth. 1969. [An identity for the schur complement of a matrix](#). *Proceedings of the American Mathematical Society*, 22(2):364–366.
- Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. 2025. [Streaming video question-answering with in-context video kv-cache retrieval](#). In *International Conference on Learning Representations*.
- Ziyang Fan, Keyu Chen, Ruilong Xing, Yulin Li, Li Jiang, and Zhuotao Tian. 2026. [FlashVID: Efficient video large language models via training-free tree-based spatiotemporal token merging](#). In *International Conference on Learning Representations (ICLR)*. Oral.
- Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiaowu Zheng, Enhong Chen, Rongrong Ji, and Xing Sun. 2024. [Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis](#). *arXiv preprint arXiv:2405.21075*.
- Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2025. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. In *International Conference on Learning Representations*.
- Xiaohu Huang, Hao Zhou, and Kai Han. 2025. [PruneVid: Visual token pruning for efficient video large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19959–19973, Vienna, Austria. Association for Computational Linguistics.
- Hyojin Jeon, Seungcheol Park, Jin-Gee Kim, and U Kang. 2023. [PET: Parameter-efficient knowledge distillation on transformer](#). *PLOS ONE*, 18(7):e0288060.
- Jihyeong Jeon, Jiwon Lee, Cheol Ryu, and U Kang. 2025. [Entity-aware generative retrieval for personalized contexts](#). In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 1078–1090. Association for Computing Machinery.
- Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song. 2025a. [Kvzip: Query-agnostic kv cache compression with context reconstruction](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Minjun Kim, Jaehyeon Choi, Jongkeun Lee, Wonjin Cho, and U Kang. 2025b. [Zero-shot quantization: A comprehensive survey](#). In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10508–10516. International Joint Conferences on Artificial Intelligence Organization. Survey Track.
- Minjun Kim, Jaehyeon Choi, Hyunwoo Yang, Jongjin Kim, Jinho Song, and U Kang. 2026a. [Prune-then-quantize or quantize-then-prune? understanding the impact of compression order in joint model compression](#). In *The Fourteenth International Conference on Learning Representations*.
- Minjun Kim, Jaeri Lee, Jongjin Kim, Jeongin Yun, Yongmo Kwon, and U Kang. 2026b. [Lampq: Towards accurate layer-wise mixed precision quantization for vision transformers](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 5691–5699.
- Alex Kulesza and Ben Taskar. 2012. [Determinantal point processes for machine learning](#). *Foundations and Trends in Machine Learning*, 5(2–3):123–286.
- Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2025. [Llava-onevision: Easy visual task transfer](#). *Transactions on Machine Learning Research*.
- Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. [BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation](#). In *International Conference on Machine Learning*, pages 12888–12900.
- Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. 2024a. [MVBench: A comprehensive multi-modal video understanding benchmark](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206.
- 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](#). In *Advances in Neural Information Processing Systems*, volume 37.
- Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. 2023. [Deja vu: Contextual sparsity for efficient LLMs at inference time](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR.
- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. [The flan collection: Designing data and methods for effective instruction tuning](#). In *Proceedings of the 40th International Conference on Machine Learning*,

- volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.
- 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, Bangkok, Thailand. Association for Computational Linguistics.
- Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. 2023. [EgoSchema: A diagnostic benchmark for very long-form video language understanding](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.
- Seungcheol Park, Jeongin Bae, Beomseok Kwon, Minjun Kim, Byeongwook Kim, Se Jung Kwon, U Kang, and Dongsoo Lee. 2025a. [Unifying uniform and binary-coding quantization for accurate compression of large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28468–28488, Vienna, Austria. Association for Computational Linguistics.
- Seungcheol Park, Hojun Choi, and U Kang. 2024. Accurate retraining-free pruning for pretrained encoder-based language models. In *The Twelfth International Conference on Learning Representations*.
- Seungcheol Park, Sojin Lee, Jongjin Kim, Jinsik Lee, Hyunjik Jo, and U Kang. 2025b. [Accurate sublayer pruning for large language models by exploiting latency and tunability information](#). In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 8213–8221. International Joint Conferences on Artificial Intelligence Organization. Main Track.
- Tairen Piao, Ikhyun Cho, and U Kang. 2022. [SensiMix: Sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for BERT compression](#). *PLOS ONE*, 17(4):e0265621.
- Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. 2024. [Streaming long video understanding with large language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 119336–119360. Curran Associates, Inc.
- Leqi Shen, Guoqiang Gong, Tao He, Yifeng Zhang, Pengzhang Liu, Sicheng Zhao, and Guiguang Ding. 2025. [FastVID: Dynamic density pruning for fast video large language models](#). In *Advances in Neural Information Processing Systems (NeurIPS)*. Poster.
- 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*, volume 235 of *Proceedings of Machine Learning Research*, pages 47901–47911. PMLR.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Senrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.
- Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo. 2021. [End-to-end dense video captioning with parallel decoding](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6847–6857.
- Yixuan Wang, Shiyu Ji, Yijun Liu, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. 2025. Lookahead q-cache: Achieving more consistent kv cache eviction via pseudo query. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshabi, and Hannaneh Hajishirzi. 2023. [Self-instruct: Aligning language models with self-generated instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.
- Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. [Finetuned language models are zero-shot learners](#). In *International Conference on Learning Representations*.
- Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. 2024a. [LongVideoBench: A benchmark for long-context interleaved video-language understanding](#). In *Advances in Neural Information Processing Systems*, volume 37.
- Shiwei Wu, Joya Chen, Kevin Qinghong Lin, Qimeng Wang, Yan Gao, Qianli Xu, Tong Xu, Yao Hu, Enhong Chen, and Mike Zheng Shou. 2024b. [Videollm-mod: Efficient video-language streaming with mixture-of-depths vision computation](#). In *Advances in Neural Information Processing Systems*, volume 37.
- Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2025. Retrieval head mechanistically explains long-context factuality. In *The Thir-*

teenth International Conference on Learning Representations.

Chaojun Xiao, Pingle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems*, volume 37.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024a. [PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3258–3270, Bangkok, Thailand. Association for Computational Linguistics.

Saelyne Yang, Sunghyun Park, Yunseok Jang, and Moontae Lee. 2024b. Ytcommentqa: Video question answerability in instructional videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Saelyne Yang, Jo Vermeulen, George Fitzmaurice, and Justin Matejka. 2024c. Aqua: Automated question-answering in software tutorial videos with visual anchors. *arXiv preprint arXiv:2403.05213*.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19792–19802.

Jeongin Yun, Jaeri Lee, Jongjin Kim, Minjun Kim, Jinho Song, and U Kang. 2026. [SharVeT: Similarity-aware parameter sharing with vector-based tuning for efficient LLM compression](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47921–47937, San Diego, California, United States. Association for Computational Linguistics.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Jinguo Zhu Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkan Yang, Chunyuan Li, and Ziwei Liu. 2025a. [Reality check on the evaluation of large multimodal models](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2135–2155, Albuquerque, New Mexico. Association for Computational Linguistics.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2025b. [LLaVA-Video: Video instruction tuning with synthetic data](#). *Transactions on Machine Learning Research*. Accepted at TMLR.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuan-dong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. 2024. [MLVU: Benchmarking multi-task long video understanding](#). *arXiv preprint arXiv:2406.04264*.

## A Uniform Budget Justification

We justify the uniform per-head KV budget  $B^{(l,g)} = B$  adopted in Section 3.2.

**Notations.** Let  $J = LG$  denote the number of layer-KV head pairs indexed by  $m \in \{1, \dots, J\}$ , and write the per-head budgets as  $b = (b_1, \dots, b_J)$  with total budget  $B_{\text{tot}}$ .

**Per-head demand.** Given a future question  $Q$ , denote  $\mathcal{T}_Q$  for the set of attending token positions of  $Q$ . For each attending position  $p \in \mathcal{T}_Q$ , the VLLM produces softmax attention weights  $\alpha_{p,i}^{(l,h)} \in [0, 1]$  from query head  $h$  to visual KV position  $i$  of layer  $l$ . Let the set  $\mathcal{H}(g)$  of query heads sharing KV head  $g$  have size  $k = H/G$ , where  $H$  is the number of query heads and  $G$  is the number of KV heads (defined in Section 2.2). We define the visual demand on head  $m = (l, g)$  as:

$$D_m(Q) = \frac{1}{|\mathcal{T}_Q| k N} \sum_{p \in \mathcal{T}_Q} \sum_{h \in \mathcal{H}(g)} \sum_{i=1}^N \alpha_{p,i}^{(l,h)}.$$

By construction,  $D_m(Q) \in [0, 1]$ . For a single question  $Q$ , the demand profile across layer-KV head pairs is generally non-uniform: questions about different temporal segments or semantic aspects of the video concentrate attention on different subsets of heads, leaving others with near-zero demand.

**Assumption 1 (i.i.d. question stream).** Let  $\pi$  denote the distribution of future questions for the video. Future questions  $Q_1, Q_2, \dots$  are sampled independently from  $\pi$ .

**Assumption 2 (equal expected demand).** Each head has a finite expected demand  $\mu_m = \mathbb{E}_{Q \sim \pi}[D_m(Q)]$ , and these expectations coincide,

$$\mu_1 = \mu_2 = \dots = \mu_J = \mu.$$

The intuition behind this assumption is as follows: although individual user questions induce non-uniform head-importance distributions, aggregating these distributions over future questions served throughout the video’s lifespan is expected to yield a uniform distribution over heads.

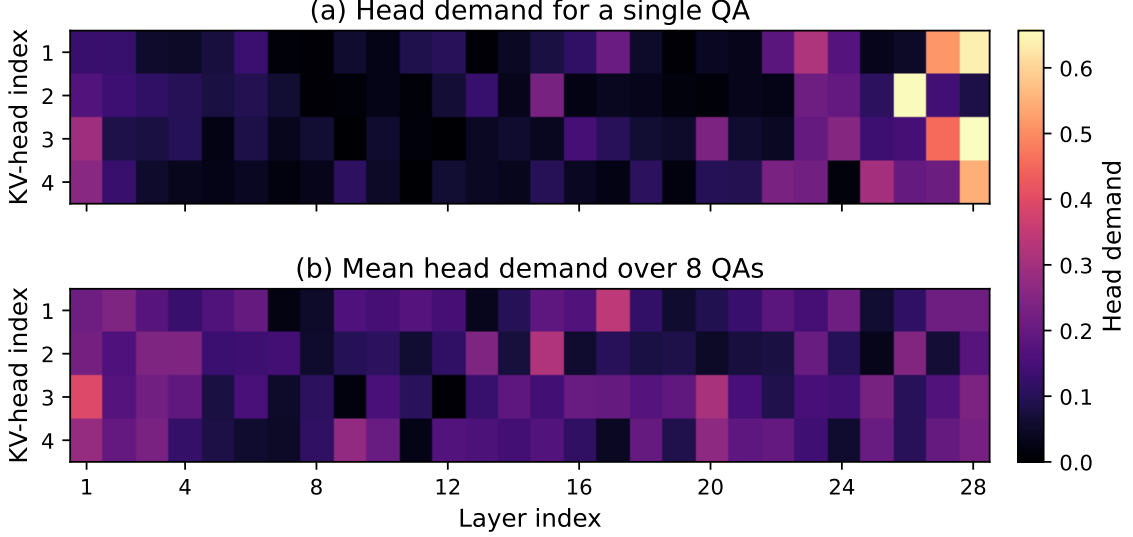


Figure 4: KV-head demand for serving (a) a single QA and (b) 8 QAs. Serving a single QA produces uneven demand concentrated on particular KV heads and layers, whereas averaging over multiple QAs makes the demand substantially more uniform across both KV heads and layers.

**Demand over the video lifespan.** Define the empirical lifetime demand on head  $m$  after  $T_L$  questions as

$$\bar{D}_m(T_L) = \frac{1}{T_L} \sum_{t=1}^{T_L} D_m(Q_t).$$

Under Assumption 1,  $D_m(Q_1), D_m(Q_2), \dots$  are i.i.d., so the strong law of large numbers gives

$$\bar{D}_m(T_L) \xrightarrow[T_L \rightarrow \infty]{a.s.} \mu_m.$$

Combined with Assumption 2,  $\mu_m = \mu$  for every  $m$ , so

$$\bar{D}_m(T_L) \xrightarrow[T_L \rightarrow \infty]{a.s.} \mu, \quad \forall m.$$

Therefore, the lifetime visual demand equalizes across heads as the question stream grows.

**Optimal budget allocation.** We instantiate budget allocation as minimizing the squared mismatch between expected demand and allocated budget,

$$\min_{b \in \mathbb{R}_{\geq 0}^J} \sum_{m=1}^J (\mu - cb_m)^2 \quad \text{s.t.} \quad \sum_{m=1}^J b_m = B_{\text{tot}},$$

where  $c > 0$  maps budget units to demand units. Form the Lagrangian

$$\mathcal{L}(b, \lambda) = \sum_{m=1}^J (\mu - cb_m)^2 + \lambda \left( \sum_{m=1}^J b_m - B_{\text{tot}} \right).$$

The stationarity condition  $\partial \mathcal{L} / \partial b_m = -2c(\mu - cb_m) + \lambda = 0$  gives

$$b_m = \frac{2c\mu - \lambda}{2c^2},$$

which is independent of  $m$ , so every coordinate of the optimum takes the same value. The budget constraint  $\sum_{m=1}^J b_m = B_{\text{tot}}$  then yields

$$b_m^* = \frac{B_{\text{tot}}}{J}, \quad \forall m.$$

We additionally provide empirical evidence to justify this uniform budget allocation. Figure 4 shows the head demand  $\bar{D}_m(T_L)$  for each layer-KV head pair  $m$ , averaged over  $T_L$  questions and measured on Qwen2.5-VL for a single video. Figure 4(a) and Figure 4(b) correspond to  $T_L = 1$  and  $T_L = 8$ , respectively. Figure 4(a) shows that head demand for a single QA is highly non-uniform across KV heads and layers, whereas Figure 4(b) shows that averaging demand over multiple QAs yields a substantially more uniform pattern across all layer-KV head pairs. Figure 4(a) is consistent with existing findings on per-question sparsity (Wu et al., 2025; Liu et al., 2023; Akhauri et al., 2024; Fu et al., 2025), while Figure 4(b) indicates that, under long-term lifetime question serving, every KV head eventually becomes equally important. This supports Assumption 2 empirically: as more questions are aggregated, per-head demand converges toward a common value, justifying the uniform per-head budget allocation derived above.

## B Proxy Temporal QA Generation

We describe how DIVE generates proxy temporal QAs from dense video captions for instruction scoring. The generation process consists of two

steps: timestamped event extraction and QA generation.

**Timestamped event extraction.** Given a video and an LLM-based dense video captioner (e.g., TEMPURA (Cheng et al., 2025)) that receives the video as input and generates textual timestamped event descriptions, DIVE extracts the timestamped description set:

$$\mathcal{E}^{\text{cap}} = \{(t_r^s, t_r^e, e_r)\}_{r=1}^R,$$

where  $R$  is the number of extracted events,  $t_r^s$  and  $t_r^e$  denote the start and end timestamps of event  $r$ , respectively, and  $e_r$  denotes its textual event description. See the example of generated descriptions in Table 6.

**QA generation.** Given  $\mathcal{E}^{\text{cap}}$ , DIVE generates a set of proxy temporal QAs

$$\mathcal{Q}^p = \{(Q_t^p, A_t^p)\}_{t=1}^T,$$

where  $T$  is the number of generated proxy QAs. To produce the QAs, DIVE applies a predefined set of rule-based QA production templates to event descriptions generated by an LLM-based dense video captioner. Specifically, DIVE instantiates a predefined set of temporal QA templates. Each template takes one or more timestamped events as input and produces a question-answer pair without using an LLM. The templates cover diverse temporal reasoning types, including before/after relations, temporal ordering, and transition reasoning. For example, Table 6 shows that predefined question skeletons such as “What happens after [event]?”, “Which happens earlier, [event] or [event]?”, and “What happens between [event] and [event]?” are instantiated by replacing the bracketed slots with the LLM-generated event descriptions.

## C KV Head-wise DPP Selection

For brevity, the remainder of this section drops the  $(l, g)$  superscript and uses  $\mathbf{L}$  for  $\mathbf{L}^{(l, g)}$  (Eq. (1)), with the understanding that the same procedure runs independently at every  $(l, g)$ . A naive approach is to enumerate all  $\binom{N}{B}$  subsets  $\mathcal{S}$  and compute  $\det(\mathbf{L}_{\mathcal{S}})$ , which is computationally prohibitive. DIVE therefore solves the problem with a greedy approach as follows. Given a partially selected subset  $\mathcal{S}_p \subseteq \{1, \dots, N\}$  with  $|\mathcal{S}_p| < B$ , DIVE evaluates the marginal gain of adding a candidate position  $i \in \{1, \dots, N\} \setminus \mathcal{S}_p$  and appends

Table 6: Examples of rule-based proxy temporal QA generation from timestamped event descriptions.

|                                                |                                                                                                                                        |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Generated timestamped event description</b> | From 0s to 4s, a person enters the room.<br>From 7s to 12s, the person picks up a cup.<br>From 15s to 20s, the person leaves the room. |
| <b>Before/After</b>                            | Q: What happens after a person enters the room?<br>A: the person picks up a cup.                                                       |
| <b>Ordering</b>                                | Q: Which happens earlier, the person picks up a cup or the person leaves the room?<br>A: the person picks up a cup.                    |
| <b>Transition</b>                              | Q: What happens between a person enters the room and the person leaves the room?<br>A: the person picks up a cup.                      |

the position with the largest gain:

$$i^* = \arg \max_{i \notin \mathcal{S}_p} [\log \det(\mathbf{L}_{\mathcal{S}_p \cup \{i\}}) - \log \det(\mathbf{L}_{\mathcal{S}_p})]. \quad (4)$$

Starting from  $\mathcal{S}_p = \emptyset$ , the procedure repeats until  $|\mathcal{S}_p| = B$ .

However, recomputing the determinant from scratch for every candidate at every step in Eq. (4) is too expensive. Using the Schur complement identity (Crabtree and Haynsworth, 1969), the determinant ratio follows:

$$\Delta_i = \frac{\det(\mathbf{L}_{\mathcal{S}_p \cup \{i\}})}{\det(\mathbf{L}_{\mathcal{S}_p})} = \mathbf{L}_{ii} - \mathbf{L}_{i, \mathcal{S}_p} \mathbf{L}_{\mathcal{S}_p}^{-1} \mathbf{L}_{\mathcal{S}_p, i},$$

Thus, the greedy step in Eq. (4) converts to  $i^* = \arg \max_{i \notin \mathcal{S}_p} \Delta_i$ .

## D Evaluation Benchmarks

We conduct experiments on the following widely-used video understanding benchmarks.

**VideoMME.** VideoMME is a comprehensive multi-modal benchmark for evaluating video understanding capabilities of VLLMs. It features 900 videos spanning 6 diverse domains and 30 subcategories, with durations ranging from 11 seconds to 1 hour. Each video is paired with high-quality human annotations, yielding 2,700 multiple-choice question-answer pairs.

**LongVideoBench.** LongVideoBench evaluates VLLMs on long-context, interleaved video-language understanding. It contains 3,763 videos with durations ranging from 8 seconds to 1 hour

and 6,678 human-annotated multiple-choice questions based on a referring-reasoning task, in which models retrieve and reason over specific multi-modal contexts referenced in the questions. The questions are categorized into 17 fine-grained types across perception and relation levels.

**MVBench.** MVBench targets temporal understanding in multi-modal video tasks, addressing the limitations of prior image-focused benchmarks. It comprises 20 systematically constructed tasks that require complex temporal reasoning, generated via static-to-dynamic transformation of static tasks.

**EgoSchema.** EgoSchema consists of approximately 5,000 five-choice multiple-choice questions derived from 250 hours of egocentric video. Each of its 289 three-minute clips requires tracking objects and actions over temporal spans 5–10 $\times$  longer than those in prior datasets, posing significant challenges for both spatial perception and extended temporal coherence.

**MLVU.** MLVU contains 3,102 multiple-choice questions across nine diverse long-video understanding tasks. Videos range from 3 minutes to 2 hours, requiring reasoning over plot, temporal order, and event retrieval, jointly testing fine-grained spatial recognition and long-range temporal reasoning.

## E Baseline Implementation

**FastV.** FastV prunes tokens at the  $K$ -th layer of the LLM using cross-modal attention scores, with a pruning ratio  $r$ . We follow the official setting with  $K = 2$ .

**VisionZip.** VisionZip prunes visual tokens at the output of the vision encoder. Because this conflicts with the pooling operations of the VLLMs we evaluate, we apply pruning after pooling. We follow the official setting by retaining dominant and contextual tokens at 54:10 ratio per frame.

**PruneVID.** PruneVID combines before-LLM compression, inner-LLM pruning during prefilling, and KV cache compression at decoding. Following the official settings, we set the threshold  $\tau = 0.8$ , the temporal segment ratio  $\gamma = 0.25$ , the token selection ratio  $\alpha = 0.4$ , and the pruning layer  $K = 10$ . The token budget is controlled via the cluster ratio  $\beta$ .

Table 7: TTFT breakdown for Qwen2.5-VL with 16 frames.

| Step                             | Time (s)    |
|----------------------------------|-------------|
| Dense captioning                 | 25.13       |
| Proxy QA generation              | 0.008       |
| Prefill                          | 0.45        |
| Instruction scoring ( $T = 16$ ) | 0.62        |
| KV head-wise DPP                 | 1.34        |
| First token generation           | 0.05        |
| <b>Total</b>                     | <b>27.6</b> |

**FastVID.** FastVID prunes visual tokens via spatiotemporal DPC-kNN. It first applies dynamic segmentation based on transition similarities, then performs frame-wise salient token selection via [CLS] attention scores, and finally compresses the remaining tokens by spatiotemporal redundancy elimination based on DPC-kNN. Following the official settings, we set the minimum number of segments  $c = 8$ , the segment threshold  $\tau = 0.9$ , the salient token ratio  $d = 0.4$ , the anchor frame step  $p = 4$ , and the merging factor  $\alpha = 0.6$ .

**FlashVID.** FlashVID compresses visual tokens via a two-step pipeline: Dynamic Temporal Segmentation (DySeg) followed by Density Spatiotemporal Pruning (STPrune). DySeg adaptively partitions the video into temporally ordered, high-redundancy segments by selecting frame transitions whose similarity falls below a threshold  $\tau$  or ranks among the  $c - 1$  most dissimilar transitions. STPrune operates within each segment, combining Density-based Token Merging (DTM) and Attention-based Token Selection (ATS) to retain both representative and salient tokens. DTM selects density-peak anchor tokens from anchor frames at a fixed interval  $p$  and merges the remaining tokens via anchor-centric aggregation with weight  $\beta$ ; ATS complements this by selecting the top salient tokens identified via [CLS]. The fraction  $d$  controls the allocation between DTM and ATS. Following the official settings, we set  $c = 8$ ,  $\tau = 0.9$ ,  $d = 0.4$ ,  $p = 4$ , and  $\beta = 0.6$ .

## F TTFT and SPT Analysis

This section analyzes DIVE’s Time-To-First-Token (TTFT) and Seconds Per Token (SPT). TTFT represents the latency of the one-time visual KV cache compression and prefill stage, while SPT represents how quickly the VLLM generates a response after prefill has completed.

Table 7 breaks down DIVE’s TTFT (Time-To-First-Token) into its constituent steps. The domi-

Table 8: SPT by retention ratio for Qwen2.5-VL with 16 frames.

| Retention ratio | FlashVID | DIVE   |
|-----------------|----------|--------|
| 20%             | 0.0349   | 0.0349 |
| 10%             | 0.0332   | 0.0332 |
| 5%              | 0.0295   | 0.0294 |

nant cost is dense video captioning (25.13s, 91.1% of total), which is performed once per video and shared across all subsequent questions. The remaining steps—prefill, instruction scoring, DPP selection, and first token generation—sum to 2.47s.

Table 8 shows SPT across methods and retention ratios. Since SPT is measured during the decoding phase after the one-time visual KV cache compression and prefill have completed, SPT depends on the size of the prefilled visual KV cache. For example, more aggressive visual KV cache compression leaves fewer visual KV pairs to attend to during decoding, thereby reducing the attention computation and lowering per-token decoding latency. Since the KV cache size is proportional to the retention ratio, SPT depends on the retention ratio rather than on the compression method itself. Table 8 confirms this directly: at a matched retention ratio, DIVE and FlashVID achieve nearly identical SPT.

Therefore, the key criterion for comparing compression methods is not how much they reduce SPT at the same retention ratio, but how well they preserve accuracy at lower retention ratios. A method that preserves accuracy under more aggressive compression operates with a smaller visual KV cache and consequently achieves lower SPT. In the lifespan question-serving setting, where a large number of questions are served, SPT is the dominant term in Cumulative Latency (CL), as defined in Section 4.1. Therefore, a method’s efficiency in this setting is determined by how much it reduces SPT without sacrificing accuracy. Per-question Latency (PL), also defined in Section 4.1, is analyzed in Section 4.4; the results in Table 3 show that PL approaches a value proportional to SPT as the number of served questions becomes sufficiently large.

## G Break-even Analysis

We analyze the break-even point at which DIVE becomes more efficient than FlashVID as the number of generated tokens per answer changes. We use the Qwen2.5-VL 16-frame latency setting in Table 2, which reports the TTFT and SPT (Seconds Per Token) of DIVE and FlashVID. Let  $T_{\text{gen}}$  denote the number of generated tokens per an-

Table 9: Break-even number of questions for DIVE against FlashVID under different generated answer lengths.

| Generated tokens $T_{\text{gen}}$ | Break-even $n^*$ vs. FlashVID |
|-----------------------------------|-------------------------------|
| 32                                | 156                           |
| 64                                | 78                            |
| 128                               | 39                            |
| 256                               | 20                            |
| 512                               | 10                            |

swer, and let  $n$  denote the number of questions asked about the same video. Here,  $\text{TTFT}_{\text{DIVE}}$  and  $\text{SPT}_{\text{DIVE}}$  denote the TTFT and SPT of DIVE, while  $\text{TTFT}_{\text{FlashVID}}$  and  $\text{SPT}_{\text{FlashVID}}$  denote those of FlashVID. Following the Cumulative Latency (CL) definition in Section 4, DIVE outperforms FlashVID when

$$\begin{aligned} & \text{TTFT}_{\text{DIVE}} + \text{SPT}_{\text{DIVE}} \times T_{\text{gen}} \times n \\ & < \text{TTFT}_{\text{FlashVID}} + \text{SPT}_{\text{FlashVID}} \times T_{\text{gen}} \times n. \end{aligned}$$

Table 9 shows that the break-even point decreases as  $T_{\text{gen}}$  increases, because DIVE’s lower per-token decoding latency is amortized over more generated tokens. When the answer is short, e.g.,  $T_{\text{gen}} = 32$ , DIVE requires many questions to offset its larger TTFT. In contrast, when the answer is longer, e.g.,  $T_{\text{gen}} = 256$  or 512, the break-even point becomes much smaller.

To estimate a realistic range of generation lengths, we additionally measure response lengths on the caption split of the LLaVA-Video-178K instruction dataset (Zhang et al., 2025b). Specifically, we tokenize responses with the Qwen2.5-VL-7B-Instruct tokenizer. Over 5,000 examples, the average response length is 236.01 tokens, with a median of 226 tokens. This suggests that the 256 token regime in Table 9 covers realistic generation lengths for video-assistant responses.

We additionally report the break-even point using PDVC (Wang et al., 2021), a lightweight captioner, instead of TEMPURA. This experiment is designed to show that using a lighter captioner further reduces the break-even point without significantly impacting accuracy. Replacing TEMPURA with PDVC reduces DIVE’s one-time TTFT from 27.6s to 6.89s, lowering the break-even point against FlashVID from  $n = 20$  to  $n = 5$  at  $T_{\text{gen}} = 256$ . Furthermore, on MLVU with LLaVA-OneVision, DIVE with PDVC-based instruction scoring achieves 58.9 accuracy, comparable to 59.1 with TEMPURA (vs. 61.3 for the full-cache vanilla model).

Table 10: Accuracy sensitivity to  $\beta$  at 5% retention. We report VMME (overall), ES (total), LVB (val), and MVB (overall).

| Model              | $\beta$ | VMME | ES   | LVB  | MVB  |
|--------------------|---------|------|------|------|------|
| LLaVA<br>OneVision | 1       | 55.7 | 59.3 | 54.6 | 56.0 |
|                    | 2       | 56.0 | 59.7 | 55.1 | 56.7 |
|                    | 4       | 55.2 | 59.2 | 54.1 | 55.6 |
| Qwen2.5-VL         | 1       | 59.3 | 56.8 | 54.9 | 67.8 |
|                    | 2       | 59.9 | 57.5 | 55.7 | 68.3 |
|                    | 4       | 59.1 | 56.6 | 55.0 | 67.3 |

## H Hyperparameter

We evaluate how the hyperparameter  $\beta$  affects accuracy. Table 10 shows that a moderate  $\beta$  achieves the best overall performance by balancing saliency and redundancy in KV-pair selection. When  $\beta = 1$ , saliency is underemphasized, thus the selected subset becomes diverse but less salient. When  $\beta = 2$ , saliency and redundancy are well balanced, yielding superior performance. When  $\beta = 4$ , the quality term dominates and the diversity is underemphasized, suggesting that the selected subset becomes less informative despite using high-saliency KV pairs.