ReSQue: Representative Surrogate Query for Streaming Video KV Cache Compression

Daehyeon Choi[1] Sangwoo Youn[1] Chanho Park[1] Jaihoon Kim[1]
Phillip Y. Lee[1] Byung-Kwan Lee[2] Minhyuk Sung[1]
[1]KAIST [2]NVIDIA

Project page: https://choidaedae.github.io/ReSQue-page/

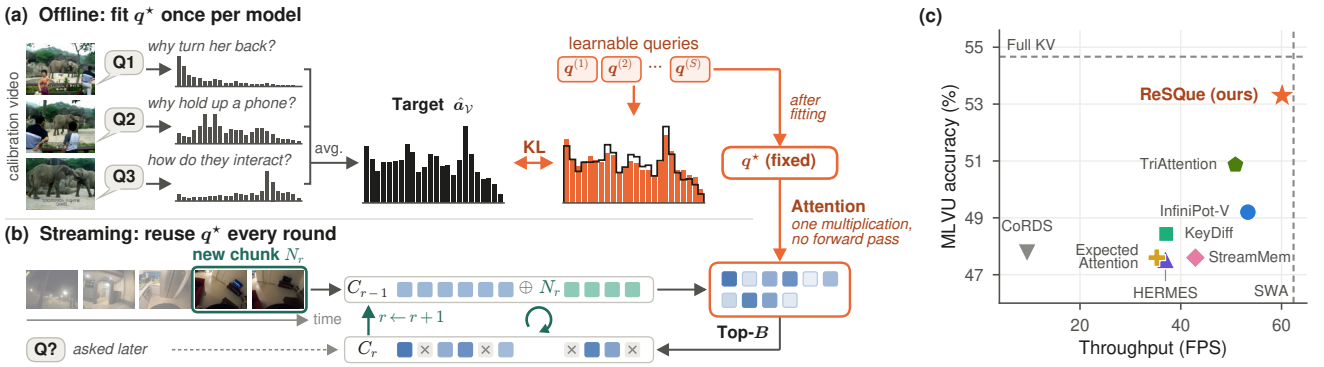


Figure 1. **Overview.** (a) ReSQue fits a few queries per attention head so that their attention matches the question-averaged attention on calibration videos. (b) During the stream, the fitted query $\boldsymbol{q}^\star$ is reused at every compression round, scoring the cache without a forward pass. (c) MLVU accuracy at 6K against throughput, the frames processed per second, averaged over three backbones. Dashed: Full KV accuracy and SWA throughput, whose accuracy lies below the plotted range.

Abstract

Streaming video understanding requires multimodal large language models (MLLMs) to compress growing KV caches before future questions are known, risking the loss of relevant information. We introduce ***ReSQue*** *(**Representative Surrogate Query**), which directly learns reusable queries to preserve information needed by future questions. We define* oracle attention *as the attention averaged over possible future questions, whose highest-scoring tokens maximize expected attention retention for a fixed cache. Since this oracle is unavailable, ReSQue constructs* target attention *by averaging question attention for each calibration video and directly optimizes head-level query vectors to match its distribution. Once learned, these queries are reused across videos and compression rounds, scoring cached keys without additional model or scorer-network forward passes. We evaluate ReSQue on three MLLM backbones and four video QA benchmarks. ReSQue achieves the best or tied-best accuracy in 25 of 27 long-video settings. Further analyses demonstrate closer alignment with future-question attention and efficient streaming compression with low recurring computational overhead.*

1. Introduction

Multimodal large language models (MLLMs) are becoming a key component of everyday AI assistants, offering flexible visual understanding through natural-language interaction [2, 52, 63]. As these assistants become more widely used, their applications extend beyond images and short clips to long-form videos, including lectures, movies, egocentric recordings, and sports broadcasts. Understanding such content requires retaining and reasoning over information that unfolds over minutes or hours, motivating recent advances in long-video understanding [23, 50, 54].

In many practical settings, however, assistants must understand video *as it unfolds*, rather than after a recording is complete. Unlike conventional long-video question answering, where the complete video and its accompanying question are provided together, *streaming video understanding* requires MLLMs to process incoming video frames as they arrive, without knowing when or what questions will be asked [7, 74]. To avoid repeatedly processing past frames, MLLMs maintain a *KV cache* containing the key and value vectors of previously processed visual tokens. As new frames arrive, the cache grows continuously, making compression necessary to stay within a bounded memory bud-

get. Streaming introduces two challenges for this compression. First, tokens must be selected without knowing what information future questions will require. Second, token selection must remain efficient as the cache evolves across repeated compression rounds. The additional computation at each round should also be limited to preserve processing throughput, measured in frames per second (FPS).

Existing approaches for streaming compression select tokens using cache statistics, as in InfiniPot-V [31], KeyDiff [46], and CoRDS [39], or attention from prompt-derived *surrogate queries*, as in StreamMem [74] and HERMES [78]. These surrogate queries are used to estimate which visual tokens future questions may attend to, guiding token selection during compression. Q-Filters [19] and TriAttention [40] leverage query statistics for accurate scoring, while Expected Attention [10] estimates future attention under a Gaussian query distribution. However, our analysis shows that the attention estimated by these methods often differs from that of actual video questions, and larger differences are associated with lower streaming QA accuracy (Sec. 6). This suggests that averaging query vectors does not generally reproduce their average attention, motivating us to directly fit surrogate queries to the attention itself. Judge Q [36] and LookaheadKV [1] instead learn soft input tokens to match future-response attention conditioned on an available request, while Fast KVzip [29] learns lightweight predictors of reconstruction-based importance. However, their learned tokens or scorers must be evaluated on each new context, introducing additional computation that can reduce processing FPS. These approaches motivate our key question: **Is recomputing surrogate queries from the evolving cache necessary for accurate streaming compression, or can directly learned, reusable queries provide comparable or better accuracy at lower cost?**

To better approximate the attention of future video questions while preserving processing throughput, we introduce **ReSQue** (**Representative Surrogate Query**), which learns reusable head-level queries for streaming KV cache compression. If future questions were known, we could score each cached visual token by the average attention it receives from those questions. We call this ideal distribution *oracle attention*. For a single selection from a fixed cache, retaining its highest-scoring tokens maximizes the attention mass preserved on average across future questions. Since future questions are unknown during streaming, we instead construct *target attention* for each calibration video by averaging the attention maps of its available questions. ReSQue directly optimizes a small set of query vectors per attention head to match these video-specific targets. Unlike statistical query proxies, these queries are explicitly fitted to video-question attention rather than derived solely from query statistics. Once learned, they can be reused across videos and compression rounds to score the evolving KV cache without an additional model or scorer-network forward pass, limiting the reduction in processing FPS.

Experiments across three MLLM backbones and both long-video and streaming question-answering benchmarks demonstrate a favorable *accuracy–throughput trade-off*. ReSQue achieves the best or tied-best accuracy in 25 of 27 long-video settings. In controlled query comparisons on held-out MLVU videos [81], ReSQue's attention more closely matches the *oracle attention* than the evaluated prompt- and statistics-based proxies. When fitted to the same target with the same attention-matching objective, directly learned queries achieve higher QA accuracy and processing FPS than soft-token and neural-scoring alternatives. Separately, controlled comparisons show that recomputing prompt queries at every compression round provides no consistent accuracy advantage over precomputing them once. On a single RTX PRO 6000 GPU, ReSQue sustains 58+ FPS on every backbone, about 96% of the throughput of a sliding-window cache (SWA) [68] on average at the same budget (Fig. 1c), while improving overall accuracy over SWA by 2.7 to 11.1 points at 6K. These results demonstrate that ReSQue improves information preservation while largely maintaining the processing throughput of simple sliding-window caching.

2. Related Work

2.1. Streaming Video Understanding

Prior work on streaming video understanding has explored training models for streaming inputs [7, 25, 60] and interaction policies [12, 49], external memories and retrieval for accessing past video content [11, 21, 23, 44, 48, 54, 64, 71, 76, 77], and techniques for reducing visual processing costs [8, 72, 75]. Under bounded memory, sliding-window approaches retain recent tokens [51, 68, 73], while other methods selectively preserve information from earlier frames [18, 32, 41, 53, 65] . In particular, InfiniPot-V [31] and CoRDS [39] select cached tokens based on redundancy and diversity, whereas StreamMem [74] and HERMES [78] use attention from text stand-in queries to guide compression. ReSQue repeatedly selects visual tokens within a fixed KV-cache budget for a frozen MLLM, preserving information for questions that may arrive later in the stream.

2.2. KV Cache Compression

KV cache compression reduces memory via quantization [24, 38], low-rank projection [5], merging [43, 58], or token eviction [80]. Most eviction methods score tokens using attention from an observation window, typically the final request-containing prompt tokens [4, 14, 16, 34, 37], or modality-aware variants [57, 59]. In streaming, the request is unavailable, leaving only recent visual queries whose attention differs from that of future questions (Sec. 6).

Query-agnostic methods instead use cache statistics [6,

9, 46] or anticipate future attention from query statistics. Q-Filters [19] and TriAttention [40] score keys using calibrated query summaries: a principal direction or a center with per-band norms. However, maximizing expected attention retention requires ranking by question-averaged attention (Sec. 4.1), which is not generally determined by simple query statistics due to softmax nonlinearity. Expected Attention [10] estimates this average with a Gaussian query model, but relies on recent context queries, inheriting the streaming limitation.

Learned methods fit scores to future attention using request-conditioned soft input tokens [1, 36] or neural importance predictors [29, 42], but require MLLM or predictor evaluation on each new cache. ReSQue combines both advantages: it learns to match question-averaged attention while using fixed head-level surrogate queries that score cached keys via a single matrix multiplication, without per-round model evaluation.

2.3. Attention as a Learning Objective

Attention has been widely used to distill Transformer models [27, 62], train retrievers from reader attention [26], and learn memory representations for streaming video generation [17]. For KV cache compression, future attention provides a training signal for learned probes [1, 36] and eviction policies [13, 22, 42]. Building on these approaches, ReSQue uses question-averaged attention from calibration videos as a learning target, directly optimizing reusable head-level queries to preserve information relevant to unknown future questions.

3. Preliminaries

We first introduce the key concepts and notation for streaming video understanding and attention-based KV cache compression, which form the basis of our method in Sec. 4.

Streaming video understanding. Given a video $\mathcal{V}$ and questions $\mathcal{U} = (U_1, \dots, U_j, \dots)$ that can arrive at any time, streaming video understanding [11, 77] aims to answer each question based on the past frames. An MLLM's vision encoder converts each incoming frame into visual tokens. Likewise, each question U_j is represented by text tokens $\mathbf{u}_j = [u_j^{(1)}, \dots, u_j^{(M)}]$, where M is the number of tokens per question.

To store information from previously processed frames for future questions, the MLLM caches the keys and values computed from visual tokens in a *KV cache*. However, as streaming video frames arrive indefinitely, the MLLM must continuously compress the KV cache, risking the loss of information needed to answer the unknown future questions. Therefore, the key challenge is to decide which visual tokens to keep while minimizing the drop in accuracy drops as little as possible.

Attention-based KV cache compression. The MLLM processes incoming frames in chunks, with each chunk-processing step called a *round* [31]. At each round, if the cache exceeds the memory budget, every attention head of every layer compresses its own cache independently. For a single head, let $\mathcal{C}$ be the set of its cached visual tokens and $\boldsymbol{K} \in \mathbb{R}^{|\mathcal{C}| \times d}$ their cached keys (with RoPE [55]); the compressor scores the tokens in $\mathcal{C}$ and keeps the B highest-scoring tokens $\mathcal{B} \subseteq \mathcal{C}$, where B is the budget.

Ideally, when a target question is given for the video $\mathcal{V}$, the queries produced by its text tokens can be used to compute attention scores for compression. Let $\boldsymbol{q}_j^{(i)} \in \mathbb{R}^d$ denote the query that the i-th text token of question U_j produces in this head. We then compute the attention map $\mathcal{A}_\mathcal{V}$ by averaging over the M queries from the question:

$$\mathcal{A}_\mathcal{V}(U_j) = \frac{1}{M} \sum_{i=1}^{M} \mathrm{softmax}\big(\boldsymbol{q}_j^{(i)} \boldsymbol{K}^\top / \sqrt{d}\big) \in \mathbb{R}^{|\mathcal{C}|}. \quad (1)$$

Each entry is the average attention weight assigned to a cached visual token by the question. The compressor uses these weights to select:

$$\mathcal{B} = \mathrm{Top}_B(\mathcal{C}; \boldsymbol{a}^*), \qquad \boldsymbol{a}^* = \mathcal{A}_\mathcal{V}(U_j), \quad (2)$$

where Top_B selects B tokens from $\mathcal{C}$ with the highest attention weights in $\boldsymbol{a}^*$. The keys and values of the selected tokens remain in the cache, while the rest are evicted to free memory. By preserving tokens with high attention weights, this compression can maintain information relevant to the target question and limit the drop in answer accuracy.

In practice, future questions are unknown at compression time, ruling out direct use of the attention scores in Eq. 2. In the following section, we introduce ReSQue, which learns a *surrogate query* to approximate these scores without access to future questions.

4. Representative Surrogate Query: ReSQue

We first identify which visual tokens should be kept to support unknown future questions (Sec. 4.1). This establishes a target attention distribution, which we approximate using calibration videos and questions (Sec. 4.2). We then examine alternative parameterizations (Sec. 4.3) and introduce ReSQue, which directly learns reusable query vectors.

4.1. Target Attention for Cache Compression

The attention-based criterion for a single question in Eq. 2 extends to the set of future questions $\mathcal{U}_\mathcal{V}$ that the same video $\mathcal{V}$ may receive:

$$\mathcal{B} = \mathrm{Top}_B(\mathcal{C}; \boldsymbol{a}^*), \qquad \boldsymbol{a}^* = \mathbb{E}_{U \sim \mathcal{U}_\mathcal{V}} \left[\mathcal{A}_\mathcal{V}(U)\right]. \quad (3)$$

Each entry of $\boldsymbol{a}^*$ is the expected attention weight assigned to one cached visual token across future questions. For a fixed cache, selecting the highest-scoring tokens maximizes

the *expected attention retention*, since the expected total attention weight of a selected set is the sum of $\boldsymbol{a}^*$ over it. In streaming, however, the cache changes as new tokens arrive and earlier ones are evicted irreversibly, so this criterion is no longer guaranteed to be optimal when applied at every round. Empirically, however, this drift does not matter: adapting the query to the video or to the stream gives no meaningful gain over a single fixed query (§6).

Since Top_B depends only on the ranking of the scores, selection is optimal for this objective when the scores rank tokens as $\boldsymbol{a}^*$ does, motivating attention distribution matching. However, computing $\boldsymbol{a}^*$ requires knowledge of the future-question distribution, which is unavailable at compression time. We therefore approximate it using calibration data, as described next.

4.2. Attention Distribution Matching

To approximate the ideal attention score $\boldsymbol{a}^*$ in Eq. 3 without access to future questions, we use a calibration dataset $\hat{\mathcal{D}}$ of videos $\mathcal{V}$ paired with finite sets of questions $\mathcal{U}_\mathcal{V}$ (implementation details in **Supplementary Document**). Attention is a distribution over the cached visual tokens of one particular video, so the target is estimated per video by averaging the attention of its calibration questions:

$$\hat{\boldsymbol{a}}_\mathcal{V} = \mathbb{E}_{U\sim\mathcal{U}_\mathcal{V}}\left[\mathcal{A}_\mathcal{V}(U)\right], \qquad (\mathcal{V},\mathcal{U}_\mathcal{V}) \in \hat{\mathcal{D}}, \tag{4}$$

where $\hat{\boldsymbol{a}}_\mathcal{V}$ estimates the target $\boldsymbol{a}^*$ of Eq. 3 for the video $\mathcal{V}$.

We use $\hat{\boldsymbol{a}}_\mathcal{V}$ as the supervision target for learning attention scores $\boldsymbol{a}_\theta(\mathcal{V})$, which are computed over the cached tokens of $\mathcal{V}$ with parameters θ shared across videos. We introduce the *attention distribution matching* objective, which minimizes the divergence Div between the target and predicted attention distributions, averaged over the calibration videos:

$$\theta^\star = \arg\min_\theta \; \mathbb{E}_{(\mathcal{V},\mathcal{U}_\mathcal{V})\sim\hat{\mathcal{D}}}\Big[\,\mathrm{Div}\left(\hat{\boldsymbol{a}}_\mathcal{V} \,\|\, \boldsymbol{a}_\theta(\mathcal{V})\right)\Big]. \tag{5}$$

Once trained, $\boldsymbol{a}_{\theta^\star}$ provides attention scores for compressing the KV cache by selecting the top-B visual tokens and evicting the rest. The behavior of this objective depends on the parameterization of $\boldsymbol{a}_\theta$, which we examine in Sec. 4.3. In practice, we use the forward KL divergence; other divergences are compared in **Supplementary Document**.

4.3. On the Choice of Parameterization

The attention distribution matching objective in Eq. 5 can be applied to different parameterizations of $\boldsymbol{a}_\theta$. We compare soft tokens, neural scorers, and directly learned query under the same target with the same objective.

Neural scorers [29, 42]. A straightforward approach is to parameterize token scores with a network f_θ that takes token representations $\boldsymbol{X}$ as input and outputs a vector of token scores, followed by softmax to produce normalized weights:

$$\boldsymbol{a}_\theta = \mathrm{softmax}\big(f_\theta(\boldsymbol{X})\big), \tag{6}$$

where $\boldsymbol{X}$ consists of key and value vectors together with token positions in KVP [42], and the input hidden states of each attention layer in Fast KVzip [29]. While this parameterization directly learns token scores, evaluating the network on incoming tokens introduces additional computation during streaming inference.

Soft tokens. Alternatively, as in Judge Q [36] and LookaheadKV [1], one can parameterize $\boldsymbol{a}_\theta$ through n learnable input embeddings $\boldsymbol{E}_\theta = [\boldsymbol{e}_\theta^{(1)}, \ldots, \boldsymbol{e}_\theta^{(n)}]$, which are appended to the cache $\mathcal{C}$ and mapped to queries by the frozen MLLM Π; $\Pi(\boldsymbol{E}_\theta;\mathcal{C})^{(i)}$ is the query that the i-th embedding produces in the head when it attends to $\mathcal{C}$:

$$\boldsymbol{a}_\theta = \frac{1}{n}\sum_{i=1}^{n} \mathrm{softmax}\big(\Pi(\boldsymbol{E}_\theta;\mathcal{C})^{(i)}\boldsymbol{K}^\top/\sqrt{d}\big). \tag{7}$$

During training, only the input embeddings are updated while the MLLM parameters remain fixed. However, the queries depend on the cache they are appended to, so they must be recomputed through the MLLM at every compression round, increasing the computational cost.

ReSQue: query vectors as parameters. We introduce ReSQue, which learns the query vectors themselves as parameters, so that neither a scorer network nor an MLLM forward pass is needed to obtain them. For each attention head, we optimize S query vectors $\boldsymbol{q}_\theta^{(1)}, \ldots, \boldsymbol{q}_\theta^{(S)}$. Since cached keys carry rotary positional embeddings (RoPE) [55] while these queries do not, we apply the rotary matrix $\boldsymbol{R}_p$ to each query, where p is the position at which a question would be appended; this holds in both training and inference. The queries produce:

$$\boldsymbol{a}_\theta = \frac{1}{S}\sum_{s=1}^{S} \mathrm{softmax}\big(\boldsymbol{q}_\theta^{(s)}\boldsymbol{R}_p\boldsymbol{K}^\top/\sqrt{d}\big). \tag{8}$$

We fit these query vectors using the attention distribution matching objective in Eq. 5. Since these queries are fixed and do not depend on the cache, scoring is one batched matrix multiplication between the query vectors and the cached keys, which takes about 0.01 s, a tenth of a compression round (Fig. 4a). In Sec. 6 (Fig. 6), we show that directly learned queries achieve the highest accuracy on all three backbones, while soft tokens and neural scorers run 1.3–1.4× and 2.1–5.2× slower, respectively.

5. Experiments

We evaluate ReSQue as a streaming compressor against prior compressors (Sec. 5.2) and measure its cost (Sec. 5.3); Sec. 6 then examines the query itself.

5.1. Setup

Models and benchmarks. We use Qwen2.5-VL-7B-Instruct [3], Qwen3-VL-8B-Instruct [2], and LLaVA-OneVision-7B [33]. We evaluate on three long-video

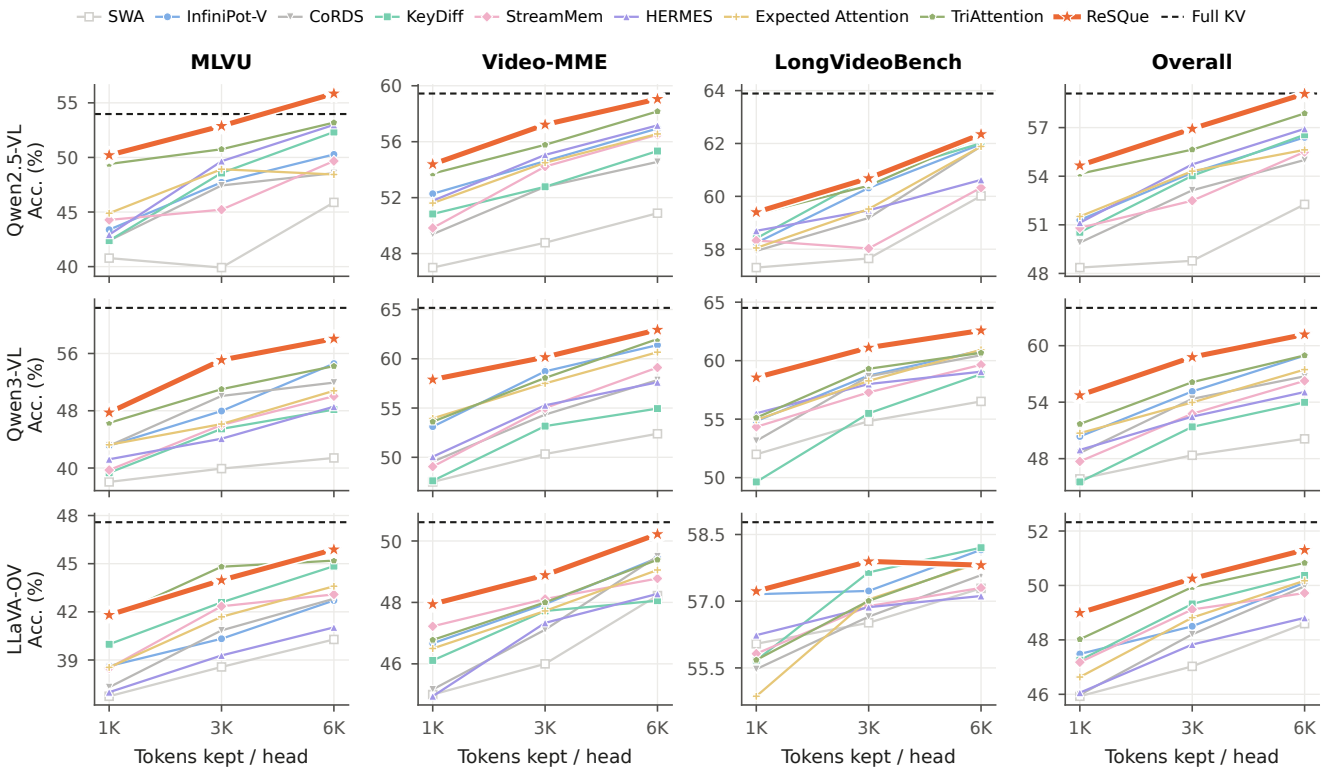


Figure 2. **Long-video question answering under streaming compression.** Accuracy against the visual tokens kept per head, with the same ReSQue query at every budget. Overall is the mean of the three benchmarks. Dashed: Full KV. Raw metrics reported in **Supplementary Document**.

question-answering benchmarks, processed as streams: MLVU [81] (502 multiple-choice questions, accuracy averaged over its holistic, single-detail and multi-detail task groups), Video-MME [15] (1,800 questions on its medium and long videos) and LongVideoBench [66] (validation set, 1,337 questions). We also evaluate on a streaming benchmark whose questions are asked during the stream, VStream-QA [77] (its ego-centric and movie splits, RVS-Ego and RVS-Movie, total 121 videos and 3,370 questions), whose open-ended answers are scored by an LLM judge.

Streaming protocol. The video is processed in chunks of 6 frames, and after each chunk the compressor reduces the cached visual tokens to a fixed budget before any question is known. On the three long-video benchmarks we test budgets of about 6K, 3K and 1K visual tokens per attention head, and on VStream-QA a budget of 6K.

Baselines. We compare with query-free compressors, InfiniPot-V [31], CoRDS [39], and KeyDiff [46]; attention-based compressors that score with a text stand-in, StreamMem [74] and HERMES [78]; and query-statistics-based compressors, Expected Attention [10] and TriAttention [40]. We also report a sliding window that keeps the system prompt and the most recent frames (SWA), as in StreamingLLM [68]. We use the official configuration for each baseline, one setting per backbone for all benchmarks and budgets, except TriAttention, whose query statistics we compute on the NExT-QA questions that ReSQue is fitted on.

Implementation details. We fit ReSQue (Eq. 5, with Div the KL divergence) on the 570 validation videos of NExT-QA [70], which share no video or question with the evaluation benchmarks, with $S = 8$ vectors per head (Eq. 8), using Muon [28] at a step size of 0.03 to 0.1. The fit takes under two hours on a single GPU (details in **Supplementary Document**).

5.2. Main Results

Offline long-video benchmarks. On the long-video benchmarks, where the question is asked after the stream has ended, ReSQue reaches the highest accuracy in 25 of the 27 settings across three backbones, three benchmarks and three budgets, two of them ties, using the same query at every budget with raw metrics reported in **Supplementary Document**. In overall accuracy, the mean over the three benchmarks, the best baseline is TriAttention at every budget and backbone. ReSQue leads it by 1.2, 1.3 and 0.5 points at 6K, 3K and 1K on Qwen2.5-VL, by 2.2, 2.7 and 3.0 on Qwen3-VL, and by 0.5, 0.4 and 1.0 on LLaVA-

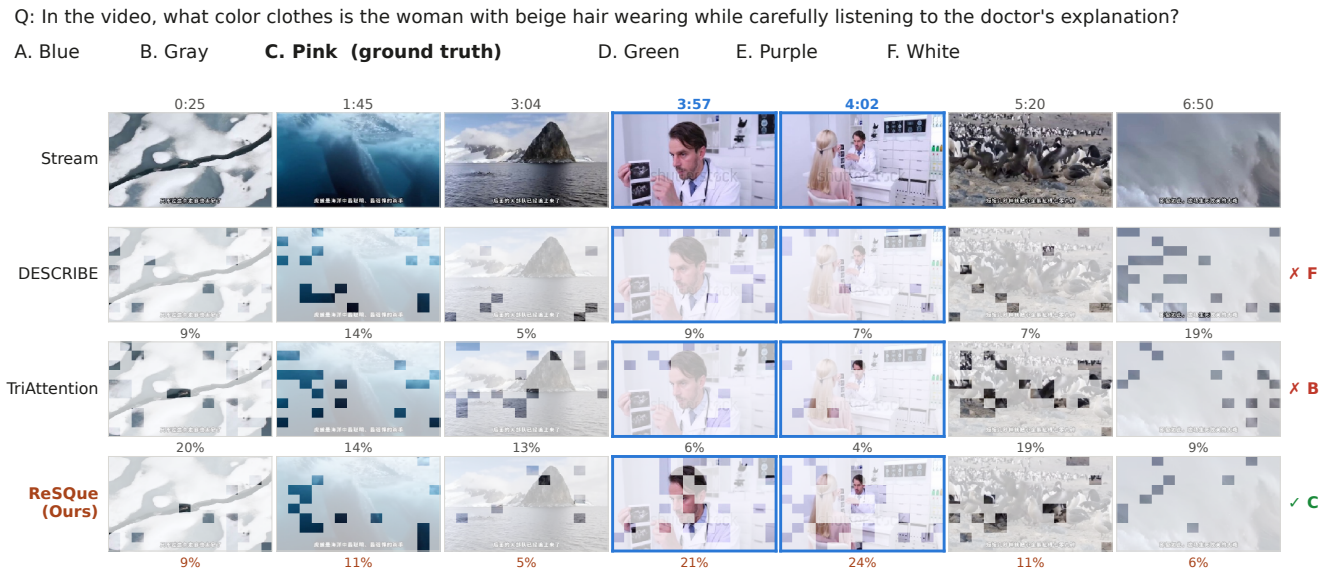


Figure 3. **Qualitative example** (MLVU, Qwen3-VL, 6K tokens per head). Patches each query keeps in one head (layer 26, KV head 6) when the question arrives, with evicted patches faded and the share kept under each frame. The blue frames lie in the evidence clip (3:55–4:05), which receives 46% of the question's attention on the uncompressed cache but spans only 2.5% of the frames. ReSQue keeps the most of the evidence frames and is the only one to answer correctly.

OV. On Qwen3-VL the margin grows as the budget shrinks, and on LLaVA-OV it is largest at 1K, when it matters most which few tokens the cache keeps. At 6K, its overall accuracy matches that of the uncompressed model on Qwen2.5-VL and is within 1.0 and 2.8 points of it on LLaVA-OV and Qwen3-VL.

TriAttention, the strongest baseline, scores keys using a trigonometric series constructed from the mean pre-RoPE query over future positional offsets, together with a query/key norm term. We compute its query statistics on the same NExT-QA questions as ReSQue, but TriAttention estimates importance from query statistics rather than directly fitting question-averaged attention (Sec. 6). Other baselines vary in relative performance across benchmarks and backbones, with neither query-free nor surrogate-query-based approaches consistently dominating. The sliding window (SWA) is the least accurate baseline in eight of the nine overall settings.

Streaming benchmark. On VStream-QA, where questions are asked while the stream is still playing, ReSQue is the most accurate compressor over all questions at the 6K budget on Qwen2.5-VL and LLaVA-OV, by 0.5 and 2.5 points, and ties TriAttention on Qwen3-VL (Table 1). The gain comes from the movie split. Since the questions are asked in real time, recent frames matter, and SWA, which keeps only those frames, is the best or second best on the ego-centric split on every backbone, ahead of ReSQue on Qwen2.5-VL and Qwen3-VL.

Qualitative example. Fig. 3 shows what three surrogate queries keep in the cache of one MLVU video, in one head, when the question arrives. ReSQue keeps more than twice as many patches of the two evidence frames as the DESCRIBE prompt and TriAttention, and only ReSQue answers correctly.

5.3. Efficiency

Compression round. A compression round encodes the new chunk, passes it through the model, and selects the tokens to keep. At 93 ms, the round of ReSQue is the closest to that of SWA (90 ms), which keeps the most recent frames without scoring any token (Fig. 4a). Encoding the chunk and the forward pass over it, which every method pays, take 84 ms of the round, and scoring with ReSQue takes

	Qwen2.5-VL-7B			Qwen3-VL-8B			LLaVA-OV-7B		
Method	Ego	Mov.	All	Ego	Mov.	All	Ego	Mov.	All
SWA	61.9	46.0	52.9	**66.6**	52.1	58.4	61.0	41.4	49.9
InfiniPot-V	60.4	49.1	54.0	65.6	53.3	58.6	60.1	41.2	49.4
CoRDS	**62.6**	48.1	54.4	65.6	54.0	59.0	60.5	41.9	50.0
KeyDiff	59.3	49.1	53.5	65.1	50.9	57.1	59.0	42.5	49.7
StreamMem	58.0	47.8	52.2	65.6	52.9	58.4	59.7	42.6	50.1
HERMES	59.5	48.2	53.1	66.5	53.4	59.1	60.7	41.4	49.8
Expected Attention	60.1	48.5	53.6	65.7	53.3	58.7	59.5	41.9	49.5
TriAttention	59.7	48.0	53.1	65.4	**55.4**	**59.7**	59.2	41.9	49.4
ReSQue	60.4	**50.6**	**54.9**	66.0	54.9	**59.7**	**61.3**	**46.0**	**52.6**

Table 1. **Streaming video QA on VStream-QA** [77], ∼6K tokens/head: accuracy (%) on its ego-centric (Ego) and movie (Mov.) splits and on all questions (All). Open-ended answers are judged correct or not by an LLM (`gpt-5-mini`). **Bold**: best; underline: second.

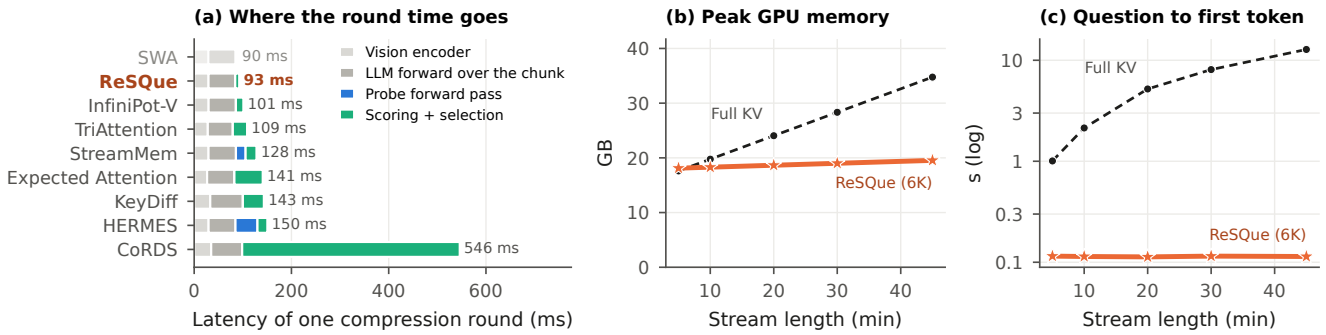


Figure 4. **Cost of streaming compression** on Qwen2.5-VL (Video-MME videos of at least 45 minutes, one RTX PRO 6000, mean over 20 videos). (a) Latency of one compression round on a 6-frame chunk at 6K, averaged over the steady-state rounds. (b) Peak GPU memory and (c) time from the question to the first token on the first T minutes of each video at 0.5 fps. Full KV prefills the whole stream after the question, and ReSQue is timed for one round plus the first token. We present results with varying backbones and budgets in **Supplementary Document**.

the remaining 8 ms because it needs no forward pass. The other compressors take 101 to 150 ms, including TriAttention (109 ms) and Expected Attention (141 ms), which also score without a forward pass, and CoRDS takes 546 ms. Since every method compresses after each 6-frame chunk, a round of t seconds lets it process $6/t$ frames per second. ReSQue can thus process 65 frames per second on Qwen2.5-VL, close to the 67 of SWA, against 55 for TriAttention and 11 for CoRDS (Fig. 1c averages the three backbones). A faster round therefore lets the compressor keep up with a stream sampled at a higher frame rate.

Memory and time to the first token. Fig. 4b,c compares ReSQue with Full KV when a question arrives, as the stream grows from 5 to 45 minutes. Full KV prefills the whole stream once the question arrives, so its time to the first token grows from 1.0 to 12.8 s and its peak memory from 17.6 to 34.8 GB. ReSQue has already processed the stream, so the first token follows the question in 0.11 s at every length, including the round for the last chunk, and its peak memory stays between 18.1 and 19.5 GB, most of it the model weights.

6. Analysis

In this section, we analyze why a single fitted query can stand in for questions it has never seen. We compare ReSQue with the queries of prior methods in how closely they follow the target attention and in accuracy, and with learned alternatives that compute their scores from the cache. Further analyses of the constant query, the calibration data and the design of the query are in **Supplementary Document**. Unless stated otherwise, ReSQue has $S = 8$ vectors per head (Eq. 8), as in deployment.

Setup. We analyze on the full MLVU benchmark [81], 349 videos and 502 questions, none used for calibration, and give three numbers for Qwen2.5-VL, Qwen3-VL and LLaVA-OV, in this order. *KL* is the forward KL divergence between the target attention $\boldsymbol{a}^*$ (Eq. 3) and the attention that a query induces over the cached visual tokens, the objective of Eq. 5 with $\boldsymbol{a}^*$ in place of $\hat{\boldsymbol{a}}_{\mathcal{V}}$, and lower is better. On an evaluation video, $\boldsymbol{a}^*$ averages the attention maps of its benchmark questions, an oracle that no compressor can compute, whereas $\hat{\boldsymbol{a}}_{\mathcal{V}}$ is the estimate that ReSQue obtains from the calibration videos. KL is measured once on the uncompressed cache and averaged over the videos, and *Acc* is the MLVU accuracy under streaming compression at the 6K budget.

Prior surrogate queries. We compare ReSQue with five surrogate queries of prior methods. Three are text queries: the DESCRIBE instruction [30, 31, 78], the assistant chat template [74], and the mean query of calibration text in TriAttention [40]. Two are vision queries, computed from the visual tokens in the cache: the queries of the most recent visual tokens, as in SnapKV [34], and their statistics, from which Expected Attention [10] estimates the attention of future queries. To isolate the effect of the query itself, in Fig. 5 each query only induces an attention distribution over the cached visual tokens by being scored against the cached keys, separated from the other components of its method (details in **Supplementary Document**); the full methods, as originally proposed, are compared in Fig. 2.

In Fig. 5, ReSQue is the closest to the target attention and the most accurate of the deployable queries on every backbone. TriAttention is the most accurate prior surrogate on every backbone, 2.7, 3.8 and 0.7 points below ReSQue. Across the six deployable queries, Acc tends to increase as KL decreases. This trend is strong on the Qwen models, with a Spearman correlation of 0.94 between the negative KL and Acc, and weaker on LLaVA-OV, with 0.49.

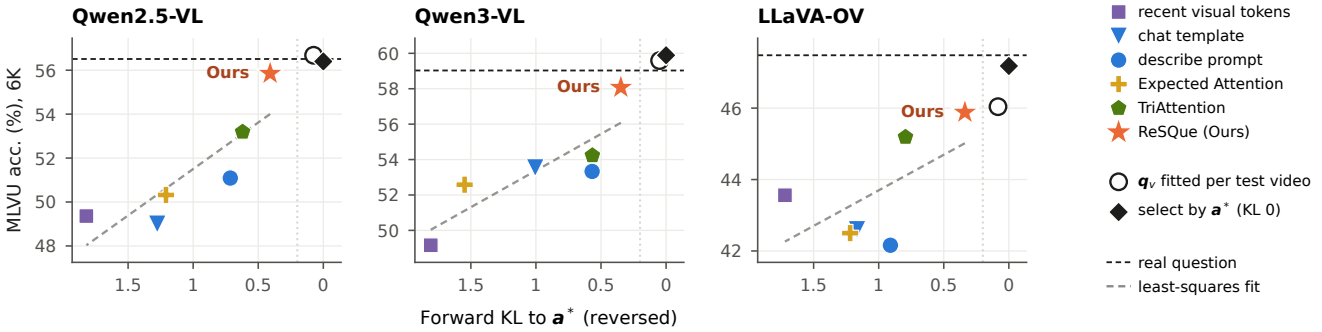


Figure 5. **Surrogate queries: closeness to the questions and accuracy.** Dashed: the real question, unknown to the compressor. Gray line: least-squares fit over the six deployable queries. Right of the dotted line: oracles that use each video's own questions.

Vision queries are far from the questions. Questions are text that reads the video, whereas a visual token attends to other visual tokens, so its query attends unlike a question. Accordingly, the vision queries are farther from the target attention than the queries of words, with a KL of 1.21 to 1.82 against 0.57 to 0.91 for DESCRIBE and TriAttention, and on the Qwen models both are less accurate than the queries of words. A query built from the video's own tokens thus cannot replace one that knows what questions look like, which is why ReSQue is fitted on video-question pairs.

Directly fitting attention outperforms query statistics approaches. TriAttention [40] and Expected Attention [10] estimate token importance from query statistics, whereas ReSQue directly fits its queries to the attention induced by video questions. In Fig. 5, both statistical approaches exhibit larger discrepancies from the distribution of real questions' attention and lower streaming accuracy than ReSQue across all three backbones. Notably, even when TriAttention uses the same calibration questions as ReSQue, directly fitting their attention yields substantially closer alignment.

Reusing learned queries improves both throughput and accuracy. ReSQue learns query vectors once during calibration and reuses them across compression rounds, directly scoring the evolving cache without recomputing the queries.

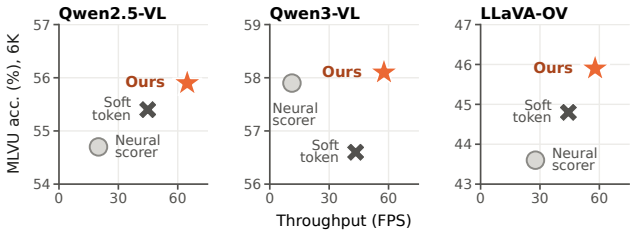


Figure 6. **Parameterization of the surrogate attention map.** MLVU accuracy at 6K against throughput, the frames processed per second (Sec. 5.3). All three are fitted with our target and objective (Eq. 5) on NExT-QA, the soft token with n=32 input embeddings.

We compare this approach with two alternatives trained using the same target attention and objective (Fig. 6): a Judge Q-like soft-token approach [36], which generates queries through the frozen MLLM at every round, and a *KVP-like neural scorer* [42], which evaluates a learned importance predictor on the current cache. ReSQue achieves 58–65 FPS, running 1.3–1.4× faster than the soft-token approach and 2.1–5.2× faster than the neural scorer, while also improving QA accuracy by 0.5–1.5 and 0.2–2.3 points, respectively. These results demonstrate that reusing directly learned queries provides both higher accuracy and greater processing throughput than the evaluated alternatives requiring additional computation at each round.

Further analyses. Additional analyses suggest that a single reusable query can effectively serve different questions, videos, and compression rounds. Selecting tokens using each video's question-averaged oracle attention $\boldsymbol{a}^*$ achieves accuracy within 0.9 points of using the actual question, while fitting a separate query $\boldsymbol{q}_v$ for each video improves over ReSQue by only 0.8, 1.5, and 0.2 points (Fig. 5). Similarly, recomputing prompt-derived queries at every round provides no consistent accuracy advantage over precomputing them once, with differences ranging from -1.6 to $+1.2$ points. Finally, all experiments use calibration data disjoint from the evaluation benchmarks, demonstrating transfer to unseen videos and questions. Calibration with video-question pairs outperforms text-only calibration, and even a small number of videos yields substantial gains (details in **Supplementary Document**).

7. Conclusion

We introduced ReSQue, which directly learns reusable head-level queries to match question-averaged attention for streaming KV cache compression. Across three MLLM backbones and four video QA benchmarks, ReSQue achieves a favorable accuracythroughput trade-off without auxiliary forward passes during compression. These results suggest that directly fitting reusable queries can provide an effective alternative to recomputing surrogate queries as the stream evolves.

LLM Usage Statement

We used Claude Opus 5.5, a large language model, to edit and revise the text of the paper for grammar and wording, and to revise the plotting code for the figures. The authors reviewed and verified all text, figures and code produced with its help and take full responsibility for the content of the paper.

References

[1] Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. LookaheadKV: Fast and accurate KV cache eviction by glimpsing into the future without generation. In *International Conference on Learning Representations*, 2026.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[4] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[5] Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

[6] Vivek Chari and Benjamin Van Durme. Compactor: Calibrated query-agnostic kv cache compression with approximate leverage scores. *arXiv preprint arXiv:2507.08143*, 2025.

[7] Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *CVPR*, 2024.

[8] Xueyi Chen, Keda Tao, Kele Shao, and Huan Wang. Streamingtom: Streaming token compression for efficient video understanding. In *CVPR*, 2026.

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

[10] Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

[11] Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video KV-cache retrieval. In *ICLR*, 2025.

[12] Xin Ding, Hao Wu, Yifan Yang, Shiqi Jiang, Qianxi Zhang, Donglin Bai, Zhibo Chen, and Ting Cao. Streammind: Unlocking full frame rate streaming video dialogue through event-gated cognition. In *ICCV*, 2025.

[13] Zican Dong, Peiyu Liu, Junyi Li, Zhipeng Chen, Han Peng, Shuo Wang, and Wayne Xin Zhao. ForesightKV: Optimizing KV cache eviction for reasoning models by learning long-term contribution. *arXiv preprint arXiv:2602.03203*, 2026.

[14] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *NeurIPS*, 2025.

[15] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. In *CVPR*, 2025.

[16] Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. In *ICLR*, 2025.

[17] Junyao Gao, Sibo Liu, Jiaxing Li, Yanan Sun, Hanxin Zhu, Weidong Zhang, Jun Zhang, and Cairong Zhao. Attn-Forcing: Learning Full-Context Attention Proxies for Streaming Video Generation, 2026.

[18] Haonan Ge, Yiwei Wang, Hang Wu, and Yujun Cai. What Should a Streaming Video Model Remember?, 2026.

[19] Nathan Godey, Alessio Devoto, Yu Zhao, Simone Scardapane, Pasquale Minervini, Éric de la Clergerie, and Benoît Sagot. Q-filters: Leveraging qk geometry for efficient kv cache compression. *arXiv preprint arXiv:2503.02812*, 2025.

[20] Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters. In *EMNLP*, 2024.

[21] Jitai Hao, Ke Yang, Di Yan, Fan Liu, Qiang Huang, and Jun Yu. ShallowStream: Index Shallow then Answer Deep for Streaming Video Understanding, 2026.

[22] Lukas Hauzenberger, Niklas Schmidinger, Anamaria-Roberta Hartl, David Stap, Thomas Schmied, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. KVpop – Key-Value Cache Compression with Predictive Online Pruning. *arXiv preprint arXiv:2607.05061*, 2026.

[23] Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *CVPR*, 2024.

[24] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In *NeurIPS*, 2024.

[25] Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: Ovbench and videochat-online. In *CVPR*, 2025.

[26] Gautier Izacard and Edouard Grave. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*, 2020.

[27] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of EMNLP*, 2020.

[28] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024.

[29] Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. Fast KVzip: Efficient and accurate LLM inference with gated KV eviction. *arXiv preprint arXiv:2601.17668*, 2026.

[30] Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot: Infinite context processing on memory-constrained LLMs. In *EMNLP*, pages 16046–16060, 2024.

[31] Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot-V: Memory-constrained KV cache compression for streaming video understanding. In *NeurIPS*, 2025.

[32] Taeyoun Kwon, Seungjin Kim, Hyeonyu Kim, and Moon Hwan Kim. DeltaS: Reading the gated linear attention state for KV cache eviction in streaming video. *arXiv preprint arXiv:2609.27470*, 2026.

[33] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *Transactions on Machine Learning Research*, 2025.

[34] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *NeurIPS*, 2024.

[35] Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. TempCompass: Do video LLMs really understand videos? In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

[36] Yijun Liu, Yixuan Wang, Yuzhuang Xu, Shiyu Ji, Yang Xu, Qingfu Zhu, and Wanxiang Che. Judge Q: Trainable queries for optimized information retention in KV cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[37] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *NeurIPS*, 2023.

[38] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *ICML*, 2024.

[39] Ailar Mahdizadeh, Puria Azadi, Muchen Li, Xiangteng He, and Leonid Sigal. CoRDS: Coreset-based representative and diverse selection for streaming video understanding. *arXiv preprint arXiv:2605.14310*, 2026.

[40] Weian Mao, Xi Lin, Wei Huang, Yuxin Xie, Tianfu Fu, Bohan Zhuang, Song Han, and Yukang Chen. TriAttention: Efficient long reasoning with trigonometric KV compression. *arXiv preprint arXiv:2604.04921*, 2026.

[41] Le Tu Ngoc Minh, Jinyeong Lim, and Dongsu Han. ProtoKV: Streaming video understanding under delayed query with summary-state memory. In *International Conference on Machine Learning*, 2026.

[42] Luca Moschella, Laura Manduchi, and Ozan Sener. Learning to evict from key-value cache. *arXiv preprint arXiv:2602.10238*, 2026.

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

[44] Zhenyu Ning, Guangda Liu, Qihao Jin, Chengwei Li, Wenchao Ding, Minyi Guo, and Jieru Zhao. LiveVLM: Efficient online video understanding via streaming-oriented KV cache and retrieval. In *DAC*, 2026.

[45] Chanho Park, Daehyeon Choi, Jihyun Lee, and Minhyuk Sung. Retrieval heads meet vision: Uncovering how vlms locate and extract visual information. In *NeurIPS*, 2026.

[46] Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. In *NeurIPS*, 2025.

[47] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *NeurIPS*, 2024.

[48] Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. In *NeurIPS*, 2024.

[49] Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video LLMs with active real-time interaction via disentangled perception, decision, and reaction. In *CVPR*, 2025.

[50] Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, Zhuang Liu, Hu Xu, Hyunwoo J. Kim, Bilge Soran, Raghuraman Krishnamoorthi, Mohamed Elhoseiny, and Vikas Chandra. Longvu: Spatiotemporal adaptive compression for long video-language understanding. In *ICML*, 2025.

[51] Yujiao Shen, Shulin Tian, Jingkang Yang, and Ziwei Liu. A simple baseline for streaming video understanding. *arXiv preprint arXiv:2604.02317*, 2026.

[52] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2026.

[53] Baiyang Song, Yuli Lin, Qiong Wu, Tao Chen, Jun Peng, Xiao Chen, Yiyi Zhou, and Rongrong Ji. Towards a dynamic and fixed-budget memory bank for efficient streaming video understanding. *arXiv preprint arXiv:2606.25658*, 2026.

[54] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, Yan Lu, Jenq-Neng Hwang, and Gaoang Wang. MovieChat: From dense token to sparse memory for long video understanding. In *CVPR*, 2024.

[55] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with

rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[56] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019.

[57] Dezhan Tu, Danylo Vashchilenko, Yuzhe Lu, and Panpan Xu. VL-Cache: Sparsity and modality-aware KV cache compression for vision-language model inference acceleration. In *ICLR*, 2025.

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

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

[60] Haibo Wang, Bo Feng, Zhengfeng Lai, Mingze Xu, Shiyu Li, Weifeng Ge, Afshin Dehghan, Meng Cao, and Ping Huang. StreamBridge: Turning your offline video large language model into a proactive streaming assistant. In *NeurIPS*, 2025.

[61] Jiahui Wang, Zuyan Liu, Yongming Rao, and Jiwen Lu. SparseMM: Head sparsity emerges from visual concept responses in MLLMs. In *ICCV*, 2025.

[62] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *NeurIPS*, 2020.

[63] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[64] Yuxuan Wang, Yiqi Song, Cihang Xie, Yang Liu, and Zilong Zheng. Videollamb: Long streaming video understanding with recurrent memory bridges. In *ICCV*, 2025.

[65] Hang Wu, Sherin Mary Mathews, Yujun Cai, Ming-Hsuan Yang, and Yiwei Wang. Semantic-aware adaptive visual memory for streaming video understanding. *arXiv preprint arXiv:2605.07897*, 2026.

[66] Haoning Wu et al. LongVideoBench: A benchmark for long-context interleaved video-language understanding. In *NeurIPS*, 2024.

[67] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *ICML*, 2008.

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

[69] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *ICLR*, 2025.

[70] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. NExT-QA: Next phase of question-answering to explaining temporal actions. In *CVPR*, 2021.

[71] Junbin Xiao, Jiajun Chen, Tianxiang Sun, Xun Yang, and Angela Yao. MuKV: Multi-grained KV cache compression for long streaming video question-answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[72] Yiweng Xie, Bo He, Junke Wang, Xiangyu Zheng, Ziyi Ye, and Zuxuan Wu. Fluxmem: Adaptive hierarchical memory for streaming video understanding. *arXiv preprint arXiv:2603.02096*, 2026.

[73] Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. StreamingVLM: Real-time understanding for infinite video streams. In *ICLR*, 2026.

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

[75] Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, et al. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025.

[76] Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, Yi Wang, and Limin Wang. StreamForest: Efficient online video understanding with persistent event memory. In *NeurIPS*, 2025.

[77] Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. Flash-VStream: Memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*, 2024.

[78] Haowei Zhang, Shudong Yang, Jinlan Fu, See-Kiong Ng, and Xipeng Qiu. HERMES: KV cache as hierarchical memory for efficient streaming video understanding. In *ACL*, 2026.

[79] Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient llms inference. In *ICML*, 2024.

[80] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023.

[81] 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. MLVU: Benchmarking multi-task long video understanding. In *CVPR*, 2025.

Supplementary Material

In this supplementary material, we first describe the implementation details of ReSQue and the evaluation protocol with the settings of every compared method (Supp. A). We then report additional results, the cost of ReSQue on all backbones and the calibration statistics of TriAttention (Supp. B), give additional analyses, namely the full numbers of the query analysis, the KL and the attention retained by a single selection and under streaming compression (Supp. C.1), why a constant query suffices (Supp. C.2) and other parameterizations of the query (Supp. C.3), and ablate the design choices of the query, the fit and the selection (Supp. D). Finally, we discuss the limitations and future directions of ReSQue (Supp. E) and report the full results (Supp. F) and additional qualitative results (Supp. G).

A. Implementation and Evaluation Details

This section gives the implementation details of ReSQue, which Sec. 4.2 defines and Sec. 5.1 evaluates. Algorithm 1 gives the two stages: the fit of the query once per model (Supp. A.1) and the streaming compression with the fitted query (Supp. A.2). It then describes the evaluation of Sec. 5 and Sec. 6: the protocol shared by all methods (Supp. A.3), the baselines of the main comparison (Supp. A.4), the queries compared in Sec. 6 (Supp. A.5) and the parameterizations compared in Sec. 4.3 (Supp. A.6).

A.1. Fitting the Query to the Target Attention

Calibration data. We fit ReSQue on the validation set of NExT-QA [70], which shares no video or question with the evaluation benchmarks: its 570 validation videos with all their questions (5,343). As the calibration question, we use the question text only: the answer options and the answer are not used. Frames are sampled as in the evaluation (Supp. A.3). Ablation results on the source and the amount of the calibration data are provided in Supp. D.4.

Target attention. Each calibration video is prefilled once into an uncompressed cache, and each of its questions is appended after the last visual token. We write p_t for the position p of Eq. 8 at time t, the position a question appended at time t takes: the next M-RoPE position after the streamed tokens on Qwen2.5-VL [3] and Qwen3-VL [2], with the same id in its temporal, height and width sections, and the next 1D position on LLaVA-OV [33]. For every layer and query head, the attention of a question over the visual tokens of this cache is the average over the question's tokens of the softmax of its queries (Eq. 1), and the target $\hat{\boldsymbol{a}}_{\mathcal{V}}$ (Eq. 4) averages it over the questions of the video.

Objective. In each head, the S vectors give $\boldsymbol{a}_\theta(\mathcal{V}) = \frac{1}{S}\sum_s \mathrm{softmax}(\boldsymbol{q}_\theta^{(s)} \boldsymbol{R}_{p_t} \boldsymbol{K}^\top/\sqrt{d})$ over the visual tokens of $\mathcal{V}$ (Eq. 8). We minimize the forward KL divergence between $\hat{\boldsymbol{a}}_{\mathcal{V}}$ and $\boldsymbol{a}_\theta(\mathcal{V})$ (Eq. 5).

Initialization and optimization. Every vector of a layer and head starts at $\bar{\boldsymbol{q}}$, the mean pre-RoPE query of the calibration questions (averaged over the tokens of each question, the questions of a video and the videos), copied to each of the S rows, plus independent random noise of relative norm 0.1 for each row. $\bar{\boldsymbol{q}}$ is one vector per head, computed once before the fit from the same question prefill as the target attention. We minimize the objective with Muon [28] optimizer, at step size 0.1 (Qwen2.5-VL) or 0.03 (Qwen3-VL, LLaVA-OV) for 2000 steps with gradient clipping at 1.0; other starting points are compared in Supp. D.3. Each step uses one video with all its questions; the checkpoint of the last step is used. The fit takes 82 / 95 / 56 minutes on one A6000 GPU for Qwen2.5-VL, Qwen3-VL and LLaVA-OV, and ReSQue has S d-dimensional vectors per layer and query head, $L \cdot H \cdot S \cdot d = 0.8$M, 1.2M and 0.8M parameters.

A.2. Inference

The lower part of Algorithm 1 gives one compression round. We write $\boldsymbol{q}^\star$ for the fitted query $\boldsymbol{q}_{\theta^\star}$ of a head, that is, its S vectors $\boldsymbol{q}_{\theta^\star}^{(s)}$.

One compression round. After a chunk is encoded, its keys and values are appended to the cache of every layer, and when the cache holds more than the budget, every layer is compressed back to B visual tokens per KV head. The system prompt is never scored or evicted, and, unlike the other compressors, ReSQue keeps no recent window: all cached visual tokens of a head compete in one ranking, with the same budget B in every KV head.

Algorithm 1 ReSQue. Fitting once per model before deployment (top) and streaming compression with the fitted query (bottom). Each query head has its own query; the query heads that share a KV head share the B visual tokens it keeps.

Fitting (once per model, before deployment)

Require: calibration videos $\hat{\mathcal{D}}$ with questions $\mathcal{U}_{\mathcal{V}}$; S vectors per head

initialize every $\boldsymbol{q}_{\theta}^{(s)}$ at $\bar{\boldsymbol{q}}$ plus noise
for each optimization step **do**
  sample $(\mathcal{V}, \mathcal{U}_{\mathcal{V}}) \in \hat{\mathcal{D}}$ and prefill $\mathcal{V}$ into an uncompressed cache
  $\hat{\boldsymbol{a}}_{\mathcal{V}} \leftarrow \mathbb{E}_{U \sim \mathcal{U}_{\mathcal{V}}} \left[\mathcal{A}_{\mathcal{V}}(U) \right]$ ▷ questions appended after the last visual token; Eq. 4
  $\theta \leftarrow$ one Muon step on $\mathrm{Div}\left(\hat{\boldsymbol{a}}_{\mathcal{V}} \,\|\, \boldsymbol{a}_{\theta}(\mathcal{V})\right)$ ▷ Eq. 5
end for
return $\theta^{\star} \leftarrow \theta$

Inference (streaming compression)

Require: chunks $\mathcal{N}_1, \mathcal{N}_2, \ldots$; budget B; fitted queries $\boldsymbol{q}_{\theta^{\star},h}^{(s)}$ of every query head h

for each chunk $\mathcal{N}_r$ **do**
  encode $\mathcal{N}_r$ attending to the cache; append its key and value vectors
  if more than B visual tokens are cached **then**
    $\boldsymbol{a}_h \leftarrow \frac{1}{S} \sum_s \mathrm{softmax}\left(\boldsymbol{q}_{\theta^{\star},h}^{(s)} \boldsymbol{R}_{p_t} \boldsymbol{K}_{g(h)}^{\top} / \sqrt{d}\right) \forall\, h$
    $g(h)$: the KV head of query head h ▷ Eq. 8
    in each KV head g, keep the B tokens with the highest $\max_{h:\, g(h)=g} \boldsymbol{a}_h$; delete the rest ▷ Supp. D.5
  end if
  if a question arrives **then** answer it from the cache
end for

Position ids after eviction. Position ids are never renumbered: the ids of the stream are computed once, as if nothing were evicted. Keys are cached after RoPE, so evicting a token only frees its slot, while the fitted queries are stored before RoPE and rotated to p_t at every round. A question arriving later also takes p_t, so the query and the question see the kept tokens at the distances of the uncompressed stream.

Scoring under grouped-query attention. In each query head, each of the S vectors $\boldsymbol{q}_{\theta^{\star}}^{(s)}$ gives a softmax over the cached visual tokens, and the score of a token is the average of these weights over the S vectors (Eq. 8). All three backbones use grouped-query attention, in which several query heads share one KV head and thus one cache, so their scores must be combined before selection. A KV head keeps the B tokens with the largest maximum score over the query heads that share it; replacing the maximum by the mean is compared in Supp. D.5, where the maximum is slightly more accurate. A round thus costs S dot products per query head and cached token, with no forward pass and no extra KV cache.

Efficient inference with a fused kernel. Since the query is fixed and a round needs no forward pass, the scoring is a regular operation on the cached keys, which we fuse: a straightforward implementation scores one head after another, whereas a single Triton kernel [56] scores all heads of a layer in parallel. The kernel fuses the rotation of $\boldsymbol{q}^{\star}$, its product with the keys, the softmax, and the per-token score, including the maximum over the query heads that share a KV head. The top-B tokens of all heads of the layer are then selected at once. Since the parallel kernel selects the same tokens as the sequential implementation exactly, it changes only the latency, not the accuracy. With the kernel, the latency of a compression round (a 6-frame chunk at the 6K budget) falls from 148 ms to 93 ms on Qwen2.5-VL (Fig. 4a), from 217 ms to 104 ms on Qwen3-VL and from 166 ms to 104 ms on LLaVA-OV (Table 4).

A.3. Common Protocol

The compressor never sees the question: it compresses the stream chunk by chunk, and a question is appended only afterwards. All methods run in one streaming pipeline with the same prompts, frame sampling and decoding, so that only the compressor differs. Full KV, the uncompressed reference, is a single prefill over all sampled frames. This protocol is shared by the main comparison, the comparison of queries and the comparison of parameterizations below, and Table 2 lists its settings for each backbone.

Frames. Frame sampling and the chunk setting follow InfiniPot-V [31]: frames are sampled uniformly over the whole video at a fixed rate (Table 2), and if this yields more frames than the maximum number each backbone supports, the video

is uniformly resampled to that maximum over its whole length. Every frame is resized to at most 1280×720 before it is encoded. The number of visual tokens per frame is kept fixed, so that a budget in frames corresponds to a fixed number of tokens.

Compression. The video is processed in chunks of 6 frames. Whenever the cache reaches the budget plus one chunk, it is compressed back to the budget, which corresponds to about 6K, 3K and 1K visual tokens per head. Compression applies to every layer unless noted in Table 3, and a method that selects tokens per head keeps the same number of tokens in every head. HERMES is the exception: it keeps the chunk size of its official code (Supp. A.4, *"Differences in cadence"*).

Questions and decoding. On the long-video benchmarks, the question is appended after the last chunk. On VStream-QA, the stream is processed up to the timestamp of each question: a chunk that contains the timestamp is cut at it, so that no method sees a frame after the question, and the question is appended there. After the question is answered, the tokens of the question and its answer are removed from the cache before the stream continues, so that later chunks and questions do not see them. When a question arrives, any cache above the budget is compressed once more, so every method answers from at most B tokens per head. Full KV caches every sampled frame and answers from the whole cache. Answers are decoded greedily with at most 300 new tokens.

Backbone	Sampling FPS	Max fr.	Tok./fr.	Budget (fr.)
Qwen2.5-VL	2	768	∼140	42 / 21 / 7
Qwen3-VL	2	768	∼140	42 / 21 / 7
LLaVA-OV	0.5	128	196	30 / 15 / 5

Table 2. **Streaming protocol of each backbone.** The budgets correspond to about 6K / 3K / 1K visual tokens per head, and the cache is compressed whenever it reaches the budget plus one 6-frame chunk.

A.4. Main Comparison

Baselines. Each baseline uses one setting per backbone for all benchmarks and budgets, taken from its public code or, where none exists, its paper. Table 3 lists the source and the configuration of each baseline; hyperparameters it does not mention take the defaults of the source.

Differences in cadence. HERMES keeps the 16-frame chunks of its official code, and StreamMem, originally run on 8-frame clips trimmed before the next clip is appended, is run on a 6-frame cadence and trims each clip in the round it arrives, in our reimplementation. The accuracies of HERMES use its 16-frame chunks, whereas its round latency in Table 4 and Fig. 4 is measured with 6-frame chunks (HERMES (6-fr.)), like every other method.

Offline long-video benchmarks. For the three long-video benchmarks, they are multiple-choice, and a prediction is correct if the first option letter in the model's answer matches the ground truth. MLVU is scored as the mean of the accuracies of its holistic, single-detail and multi-detail task groups. Video-MME is scored on its medium and long videos, 900 questions each and without subtitles, as the mean of the two accuracies. LongVideoBench is scored on its validation set as the mean over its four video-length groups (15, 60, 600 and 3600 seconds).

Streaming benchmark. On VStream-QA [77], open-ended answers are judged by `gpt-5-mini` with the judging prompt and protocol of the official InfiniPot-V implementation (`rvs_judge.py`), which follows the protocol of Flash-VStream [77]; accuracy is the fraction of answers the judge marks as correct.

A.5. Query Comparison

This subsection details the queries compared in Sec. 6. Unlike the main comparison, which runs each baseline as proposed, this comparison isolates the query: each query is only scored against the cached keys, with the selector, the streaming protocol and the budgets of Supp. A.3 held fixed, and every other component of the original method is left out. The accuracy of every query at the three budgets is in Table 18 (Supp. F).

Table 3. **Baseline configurations**, identical on every benchmark and budget.

Method	Source	Configuration
SWA [68]	reimplemented	System prompt and the most recent frames up to the budget; no scoring.
InfiniPot-V [31]	official code (github.com/aiha-lab/InfiniPot-V)	Half of the budget by the temporal-redundancy score against the most recent 25% of the cached frames (always kept), the rest by value norm; with layer-wise adaptive pooling.
CoRDS [39]	official code (github.com/ailarmhz/CoRDS)	Coreset selection on the bottom 25% of layers; the other layers reuse the indices of the nearest selected layer.
KeyDiff [46]	KVPress (github.com/NVIDIA/KVPress)	Negative similarity of each key to the mean key; the most recent 25% of the cached frames always kept.
StreamMem [74]	reimplemented	Chat-template proxy query; evicted tokens merged frame-wise by their weights, with YaRN position scaling and input-frame filtering.
HERMES [78]	official code (github.com/haowei-freesky/HERMES)	Shallowest 10% of layers keep recent tokens, deepest 30% score by attention, the rest by a 'local' and a 'global' probe; 16-frame chunks; evicted tokens mean-pooled into one slot, with RoPE fold.
Expected Attention [10]	KVPress (github.com/NVIDIA/KVPress)	Gaussian over the queries of the newest chunk, expected attention over 512 future positions, with the value norm; ϵ=0.02, mean over the query heads of a KV head.
TriAttention [40]	official code (github.com/WeianMao/triattention)	Full score, the trigonometric term plus the norm term, with query statistics computed on the NExT-QA calibration questions that ReSQue is fitted on, namely the mean pre-RoPE query of Sec. 6 and the mean norm of each RoPE frequency pair, and future offsets $1, 2, 4, \ldots, 2^{16}$.

Queries. Besides the DESCRIBE instruction of the main text, we include EXPLAIN and SUMMARIZE, two generic prompts of the same kind, to show that the conclusions drawn from DESCRIBE do not depend on its particular wording.

- **DESCRIBE**: "`Describe the video.`"
- **EXPLAIN**: "`Explain what you see.`"
- **SUMMARIZE**: "`Summarize the video.`"
- **Chat template**: the generation-prompt suffix of the model's chat format, `<|im_start|>assistant`, as in StreamMem [74].
- **Recent visual tokens**: the queries of the 64 most recent visual tokens of the cache, as in SnapKV [34].
- **Q-Filters** [19]: the first right singular vector of 3,000 queries of the visual tokens of the NExT-QA calibration videos, per query head, averaged over the heads of a KV head, as in the original recipe; keys are scored by their inner product with this filter, without a softmax.
- **Expected Attention** [10]: the mean and the covariance of the queries of the visual tokens of the newest chunk, with the expected attention to each key computed as in the original method but without the value-norm factor.
- **TriAttention** [40]: its full score, the trigonometric term plus the norm term. The trigonometric term is the inner product between the cached key and the mean pre-RoPE query of the NExT-QA calibration questions, rotated to future positions $1, 2, 4, \ldots, 2^{16}$ ahead of the question position; the norm term is $\sum_f (\mathbb{E}|q_f| - |\bar{q}_f|)\,|k_f|$ over the RoPE frequency pairs f.

Each text query is rotated to p_t (Supp. A.1), and each prompt token gives one query vector. We omit the KL of Q-Filters because it produces unnormalized importance scores rather than a native attention distribution, and report only its accuracy (Table 18).

Attention induced by each query. These distributions are the ones whose KL to $\boldsymbol{a}^*$ is reported in Fig. 5. For every head, a surrogate query is a set of query vectors, each rotated to p_t, and it induces an attention distribution over the cached visual tokens.

- **ReSQue**: the mean of the S softmax maps of Eq. 8.
- **DESCRIBE, EXPLAIN, SUMMARIZE, chat template**: the mean of the M (number of tokens) softmax maps of the query of each prompt token, as in Eq. 1, computed on the uncompressed cache.

- **Recent visual tokens**: the mean of the softmax maps of the queries of the 64 most recent visual tokens.
- **Expected Attention**: the softmax over cached tokens of the attention that keys receive in expectation under the Gaussian model of the queries of the visual tokens of the newest chunk.
- **TriAttention**: the softmax over cached tokens of its full score (trigonometric term plus norm term), scaled by $1/\sqrt{d}$ like the logits of attention.

A.6. Parameterization Comparison

This subsection details the parameterizations compared in Sec. 4.3. All are fitted on the 570 NExT-QA validation videos with the target and objective of Eq. 5, and evaluated on MLVU at 6K with the selector and protocol of Supp. A.3.

Soft tokens (Judge Q-like). The n learned input embeddings are appended after the cache at p_t and passed through the frozen model at every compression round; the query that each embedding produces in each head scores the cached keys. Fig. 6 uses $n = 32$, the default of the official Judge Q implementation [36]. The embeddings are initialized from the mean input embeddings of the NExT-QA question tokens plus noise, then trained for 2,000 steps with Muon, and the final checkpoint is used. Other numbers of embeddings are compared in Supp. C.3.

Neural scorers (KVP-like). One network per (layer, KV head) reads each cached visual token, namely its key, de-rotated to p_t, and bounded features of its position (time, row and column), and outputs one score per query head of the group. Fig. 6 uses the MLP, with 0.85M parameters on Qwen2.5-VL. The scorer is trained alone with Muon for 2,000 steps, and the final checkpoint is used. It reads the cached keys rather than the hidden states that Fast KVzip [29] uses, a cache-only setting shared by all parameterizations; Supp. C.3 discusses this choice and compares architectures and inputs.

B. Additional Results

This section extends the experiments of Sec. 5 with the cost of ReSQue on all backbones (Supp. B.1) and the calibration statistics of TriAttention in the main comparison (Supp. B.2).

B.1. Efficiency on All Backbones

This subsection extends the efficiency analysis of Sec. 5.3 to all three backbones.

Protocol. Table 4 follows the protocol of Fig. 4: one RTX PRO 6000, and 22 Video-MME videos of at least 45 minutes, of which the first two are discarded as warm-up and the other 20 averaged. A round is timed in the steady state, excluding the first and the last chunk and the rounds before the cache is full, and FPS is the 6 frames of a chunk divided by its round time. The answer time and the peak memory at the bottom of the table use Video-MME medium and long videos, so that all three backbones are measured on the same videos. The peak memory includes the model weights, which take 15.5, 16.3 and 15.0 GB on the three backbones, so the percentages are reductions of the total memory, not of the cache. The answer time runs from the start of the question prefill to the end of the decoded answer, a single letter. Full KV is the offline, memory-unconstrained setting of Table 13: it prefills all visual tokens after the question, whereas ReSQue answers from the cache it compressed during the stream. Both are measured with every video resampled to about 50K visual tokens.

Results. A round of ReSQue takes 0.093 to 0.104 s (58 to 65 FPS), within 0.006 s of SWA, which scores no token. The scoring of ReSQue is a single matrix product of its fixed query with the keys, which makes it easy to fuse with the selection into one kernel (Supp. A.2); the latencies of ReSQue in (a) are measured with it, and without it a round takes 0.148, 0.217 and 0.166 s on the three backbones (Table 4(b)). TriAttention is measured in (a) with its head-batched selection. The fused selection shortens a round of ReSQue by 37 to 52% at 6K, by 24 to 37% at 3K and by 25 to 40% at 1K, and a round of TriAttention by 14 to 34% at 6K, and by at most 26% at 3K and 1K, with no gain on LLaVA-OV at 3K. In (b), the per-head rows run the selection as a loop over heads, whereas the fused rows select with the head-batched selection of TriAttention and the fused kernel of ReSQue. The answer time is 130 to 170 times shorter with ReSQue than with offline Full KV; this compares a streaming and an offline setting, not the speed of the two per token. ReSQue has compressed the stream before the question arrives, so only the 6K cache is read when it is asked, whereas Full KV prefills all 50K visual tokens after the question. Its peak memory is 36 to 47% lower. Most of the remaining memory is the model weights: the 6K cache takes 0.3 to 0.8 GB against 2.7 to 6.9 GB for 50K tokens, so lowering the budget to 1K saves little more. The latency of CoRDS falls most with the budget (0.546, 0.320 and 0.149 s at 6K, 3K and 1K on Qwen2.5-VL), because its selection grows with the

method	Qwen2.5-VL				Qwen3-VL				LLaVA-OV			
	6K (s)	FPS	3K (s)	1K (s)	6K (s)	FPS	3K (s)	1K (s)	6K (s)	FPS	3K (s)	1K (s)
(a) Round latency (s) at 6K, 3K and 1K, and frames per second at 6K												
SWA	0.090	67.0	0.090	0.088	0.098	61.4	0.091	0.086	0.102	58.8	0.097	0.093
InfiniPot-V	0.101	59.3	0.092	0.089	0.121	49.5	0.101	0.095	0.117	51.1	0.111	0.105
KeyDiff	0.143	41.9	0.132	0.129	0.217	27.7	0.168	0.162	0.144	41.8	0.125	0.127
CoRDS	0.546	11.0	0.320	0.149	1.192	5.0	0.582	0.256	0.481	12.5	0.271	0.166
StreamMem	0.128	46.8	0.135	0.127	0.154	38.8	0.151	0.152	0.139	43.1	0.129	0.124
HERMES (6-fr.)	0.150	39.9	0.156	0.146	0.178	33.7	0.173	0.177	0.161	37.3	0.158	0.150
Expected Attention	0.141	42.6	0.144	0.140	0.239	25.1	0.180	0.169	0.158	38.1	0.136	0.138
TriAttention	0.109	55.1	0.108	0.107	0.122	49.3	0.111	0.105	0.125	48.1	0.116	0.111
ReSQue	**0.093**	**64.8**	**0.088**	**0.084**	**0.104**	**57.7**	**0.093**	**0.089**	**0.104**	**57.9**	**0.101**	**0.097**
(b) Round latency (s) with the per-head selection and with the fused selection, and frames per second at 6K												
TriAttention (per-head)	0.143	42.0	0.113	0.110	0.186	32.3	0.142	0.141	0.145	41.4	0.114	0.123
TriAttention (fused)	0.109	55.1	0.108	0.107	0.122	49.3	0.111	0.105	0.125	48.1	0.116	0.111
ReSQue (per-head)	0.148	40.7	0.117	0.115	0.217	27.7	0.148	0.149	0.166	36.1	0.133	0.129
ReSQue (fused)	**0.093**	**64.8**	**0.088**	**0.084**	**0.104**	**57.7**	**0.093**	**0.089**	**0.104**	**57.9**	**0.101**	**0.097**
(c) Peak memory (GB) / answer time (s): streaming (ReSQue) vs. offline (Full KV)												
Full KV (50K)	26.0 / 6.563				31.0 / 6.340				29.8 / 4.997			
ReSQue (6K)	16.7 / 0.038				18.4 / 0.049				15.8 / 0.039			
ReSQue (3K)	16.6 / 0.040				18.0 / 0.045				15.6 / 0.042			
ReSQue (1K)	16.5 / 0.035				17.8 / 0.046				15.5 / 0.035			

Table 4. **Cost of streaming compression.** (a) Latency in seconds of a 6-frame compression round at the 6K, 3K and 1K budgets and, at 6K, the FPS it allows (= 6/latency); SWA scores no token (gray). (b) The same for TriAttention and ReSQue with the per-head selection and with the fused selection; its fused rows are those of (a). (c) Peak memory / answer time of offline Full KV and of streaming ReSQue at 6K, 3K and 1K; the answer times compare a streaming with an offline setting, since ReSQue compresses the stream before the question arrives. The latencies at 3K and 1K and the memory of ReSQue at 3K were measured separately with the same protocol.

Calibration statistics	Qwen2.5-VL	Qwen3-VL	LLaVA-OV
Wikipedia text (no video)	49.7	49.8	44.3
NExT-QA questions (main comparison)	53.2	54.2	45.2

Table 5. **TriAttention with different calibration statistics.** MLVU accuracy (%) at the 6K budget of TriAttention with statistics computed on Wikipedia text and on the NExT-QA questions, as in the main comparison (Table 13).

number of kept tokens. A round of ReSQue takes 0.088 to 0.101 s at 3K and 0.084 to 0.097 s at 1K, within 0.004 s of SWA, which scores no token.

B.2. TriAttention with Other Calibration Statistics

The main comparison computes the statistics of TriAttention on the NExT-QA questions that ReSQue is fitted on (Supp. A.4), so that the two methods see the same calibration data. TriAttention itself calibrates on generic text, without any video or question. This subsection shows how much that choice matters for TriAttention, with Table 5.

Calibration statistics. In both settings the statistics are the mean pre-RoPE query of every layer and query head, and the mean norm of each RoPE frequency pair, taken over the query tokens and weighted by token; each query is de-rotated at its own position, and TriAttention has no fit. *Wikipedia text* uses generic text without any video or question, from the Wikipedia source of the calibration-source experiment (Supp. D.4). *NExT-QA questions* are the tokens of the questions of the NExT-QA pool of ReSQue, computed with their video in the context.

The main comparison gives TriAttention its more favorable calibration. The statistics of the questions are more accurate than those of the text on all three backbones (Table 5), most clearly on the two Qwen models. With text statistics, TriAttention would fall more than twice as far below ReSQue on every backbone, so the comparison of Table 13 gives TriAttention the more favorable of the two calibrations. As for the calibration of ReSQue (Supp. D.4), questions asked about

	Qwen2.5-VL				Qwen3-VL				LLaVA-OV			
Query	KL	Single	Stream	Acc	KL	Single	Stream	Acc	KL	Single	Stream	Acc
Recent visual tokens	1.82	0.630	0.559	49.4	1.81	0.724	0.690	49.2	1.72	0.851	0.754	43.6
Chat template	1.28	0.674	0.638	49.0	1.01	0.802	0.756	53.5	1.17	0.869	0.770	42.6
DESCRIBE	0.71	0.703	0.655	51.1	0.57	0.826	0.780	53.3	0.91	0.885	0.783	42.2
Expected Attention	1.21	0.637	0.538	50.3	1.55	0.728	0.625	52.6	1.22	0.853	0.749	42.5
TriAttention	0.62	0.701	0.560	53.2	0.57	0.828	0.700	54.2	0.79	0.896	0.788	45.2
ReSQue	**0.41**	**0.730**	**0.659**	**55.9**	**0.35**	**0.847**	**0.786**	**58.1**	**0.34**	**0.914**	**0.800**	**45.9**
Question known	–	0.796	0.715	56.5	–	0.897	0.821	59.0	–	0.940	0.818	47.5
Video's questions known	0	0.786	0.713	56.4	0	0.892	0.819	59.9	0	0.935	0.817	47.2
$\boldsymbol{q}_{\mathcal{V}}$ fitted per test video	0.07	0.778	0.700	56.7	0.05	0.887	0.816	59.6	0.08	0.933	0.816	46.0
Spearman with Acc	+0.94	+0.77	+0.49	–	+0.94	+0.94	+0.71	–	+0.49	+0.49	+0.60	–

Table 6. **KL, expected attention retention and accuracy of each query** on MLVU (349 videos, 502 questions). KL: to $\boldsymbol{a}^*$, on the uncompressed cache. Single, Stream: expected attention retention of one selection on the uncompressed cache and after the last streaming round. Acc: streaming compression at the 6K budget. Rows below ReSQue use the questions. Last row: Spearman correlation with Acc over the six queries (−KL for KL). Bold: best of the six queries.

videos are a better calibration source than generic text.

C. Additional Analysis

This section extends the analysis of Sec. 6: the full numbers of the query analysis (Supp. C.1), why a constant query suffices (Supp. C.2), and alternative parameterizations of the query (Supp. C.3).

C.1. Query Analysis: KL and Attention Retention

This subsection gives the numbers behind the query analysis of Sec. 6 and compares, for the same queries, the attention retained by a single selection on the uncompressed cache with the attention retained under streaming compression. It then discusses which changes of the token scores between rounds can account for the difference between the two.

Measures. Table 6 reports three measures for each query. The *KL* is the forward KL divergence between the target attention $\boldsymbol{a}^*$ (Eq. 3) and the attention that a query induces over the cached visual tokens, measured once on the uncompressed cache and averaged over the 349 MLVU videos; with the accuracy, it is the content of Fig. 5, and the accuracy is that of MLVU under streaming compression at the 6K budget. KL is measured on the uncompressed cache, whereas the compressor works on a cache that earlier rounds have thinned, so we also measure the attention that the kept tokens hold in both settings, with the *expected attention retention*. For a kept set $\mathcal{B}_g$ of a KV head g and a question U with attention $\mathcal{A}_g(U)$ on the uncompressed cache, normalized to sum to one over the cached visual tokens,

$$\mathrm{ExpRet}_g(\mathcal{B}_g) = \mathbb{E}_U\Big[\sum_{c\in\mathcal{B}_g} \mathcal{A}_g(U)^{(c)}\Big] = \sum_{c\in\mathcal{B}_g} \boldsymbol{a}_g^{*(c)},$$

the share of the attention of a question that the kept tokens hold, on average over questions. We estimate it with the questions of each video and average it over the KV heads of all layers and over the videos. It is below one even for the best choice of B tokens: the best B tokens of each question hold 0.80, 0.90 and 0.94 of its attention on the three backbones. We report it for two kept sets of each query: **Single**, one selection on the uncompressed cache, and **Stream**, the kept set after the last round of the streaming run, where each round scores the cache that earlier rounds left. The queries are those of Fig. 5; DESCRIBE, recent visual tokens, the chat template and Expected Attention are computed on the stream at every round (computing DESCRIBE once on the calibration videos instead changes its attention retention by at most 0.010 after streaming). Two oracles that know the questions serve as references: *Question known*, where each question compresses the stream on its own and probes with its text at every round, and *Video's questions known*, which probes with all the questions of the video. $\boldsymbol{q}_{\mathcal{V}}$ is a query fitted to the target attention $\boldsymbol{a}^*$ of each test video.

ReSQue is the closest to the target attention and the most accurate query. Across the six queries, a lower KL goes with a higher accuracy, strongly on the Qwen models and more weakly on LLaVA-OV (Table 6), and ReSQue has the lowest KL and the highest accuracy on all three backbones. It is the only one of these queries that is fitted directly to the attention of real questions (Sec. 6).

Single and Stream. Stream is below Single for every query and for both oracles, by 0.03 to 0.14 for the queries and by 0.07 to 0.12 for the oracles. The difference is not specific to the surrogate query: the oracles that know the questions show the same order of magnitude, and the difference of ReSQue (0.06 to 0.11) is within the range of the other queries. The order of the queries is largely kept: the rank correlation between Single and Stream over the six queries is 0.89, 0.77 and 0.94. ReSQue holds the most attention of the six queries on all three backbones, both in a single selection and after streaming. After streaming, ReSQue holds 0.92, 0.96 and 0.98 of the attention held by *Question known*. On the Qwen models, the attention retention after streaming tracks the accuracy less closely than the KL does (Table 6, last row).

The drop comes from scores that change between rounds. If the score of every token were fixed, keeping at each round the top B of the kept tokens and the new chunk would give exactly the top B of the whole stream, since a token kept at the end of any round is among the top B of the prefix by that score; the irreversible eviction would then have no effect. A constant query vector has this property when the score of a token depends only on the token. We verify it on the scores of the uncompressed cache: a selection simulated chunk by chunk with these scores holds exactly the attention of the Single selection, for every query. The difference between Single and Stream therefore comes from the scores changing between rounds. Three changes of the scores are possible.

- **RoPE drift.** The query is rotated to the position at which a question would be appended, which moves with every chunk, so its distance to each cached token differs between rounds and from that of a question at the end of the stream.
- **Key drift.** The keys of a new chunk are computed on a cache that earlier rounds have already compressed, so they differ from the keys on the uncompressed cache on which the target attention is defined.
- **Normalization.** The score of a token is the average of S softmax maps, each normalized over the tokens currently in the cache, and a KV head takes the maximum of these scores over the query heads that share it (Supp. D.5). The normalizers change as the cache is thinned and differ between the query heads, so the ranking of the tokens can change between rounds even for a fixed query; only for one vector in a single query head is the softmax monotone in the logit and the ranking unchanged.

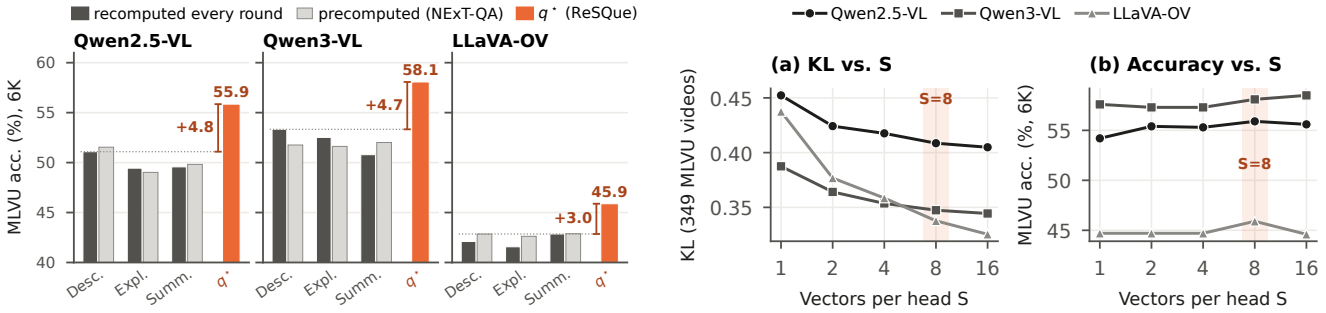
We do not separate their contributions here. The same changes act on the oracles, which also hold less attention after streaming than in a single selection, so they are properties of the streaming procedure rather than of a particular surrogate query.

C.2. Why a Constant Query Vector Suffices

This subsection extends the analysis of Sec. 6. ReSQue is a constant vector fixed before deployment, and we ask whether it holds up as the questions, the videos and the stream vary, in KL to $\boldsymbol{a}^*$ and in MLVU accuracy.

One query for all questions of a video is nearly as accurate as the real question. ReSQue targets $\boldsymbol{a}^*$, the average attention of the questions of a video, since the questions are unknown when the stream is compressed; it approximates $\boldsymbol{a}^*$ by fitting to the $\hat{\boldsymbol{a}}_{\mathcal{V}}$ of the calibration videos. Selecting by $\boldsymbol{a}^*$, an oracle that uses each video's own questions, is within 0.9 points of compressing with the real question (Fig. 5), although every question of a video is then answered from the same tokens. The questions of a video attend alike: since $\boldsymbol{a}^*$ is their average, the KL of any query from the real questions is their spread around $\boldsymbol{a}^*$ plus the query's own KL from $\boldsymbol{a}^*$, and on the 70 MLVU videos with more than one question this spread is only about a quarter of the distance of ReSQue from the real questions. The rest is the distance of ReSQue from $\boldsymbol{a}^*$, which a better constant query could still reduce.

A per-video query matches the attention far better but improves the accuracy little. A query fitted to each video's own $\boldsymbol{a}^*$, $\boldsymbol{q}_{\mathcal{V}}$ with S vectors per head and the objective of Eq. 5, is an oracle that needs the video's questions, so no deployable method can compute it. It lowers the KL to $\boldsymbol{a}^*$ from 0.34–0.41 (ReSQue) to 0.05–0.08, but raises the accuracy by only 0.8, 1.5 and 0.2 points on the 349 MLVU videos (Fig. 5). Fitting each video separately thus adds little accuracy beyond one query shared across videos.



(a) **Recomputing the prompt query.** MLVU accuracy at 6K with the query of a text prompt recomputed at every round (dark) or precomputed once on NExT-QA (light). Δ: gain of ReSQue, also a constant, over the best recomputed prompt.

(b) **Vectors per head.** Effect of the number of query vectors per head S on the attention divergence (left, KL on the 349 MLVU videos) and the MLVU accuracy at the 6K budget (right); ReSQue uses S=8 (shaded).

Figure 7. **Additional analysis.** (a) Recomputing the prompt query. (b) Effect of the number of query vectors per head.

Recomputing the prompt query at every round brings no consistent gain. Prompt-based compressors recompute the query of their text prompt from the current cache at every round, so that it can adapt to the video seen so far. Replacing it with a constant, the query of the same prompt computed in advance on the NExT-QA calibration videos, averaged and used at every round without a forward pass, is neither better nor worse (Fig. 7a): across the DESCRIBE, EXPLAIN and SUMMARIZE prompts (Supp. A.5) and three backbones, the precomputed query is more accurate than the recomputed one in some settings and less accurate in others, with differences from -1.6 to $+1.2$ points and no consistent direction. Which query is used matters far more: ReSQue, also a constant, is 3.0 to 4.8 points more accurate than the best of the three recomputed prompts (Fig. 7a, Δ). This concerns the three generic prompts evaluated here; it does not show that learned probes, such as the soft tokens of Judge Q [36] and LookaheadKV [1], need no recomputation.

C.3. Alternative Parameterizations

This subsection extends the comparison of parameterizations of Sec. 4.3. Fig. 6 compares ReSQue with one configuration of a neural scorer and one configuration of soft tokens. Here we vary the main design choices of each alternative to test whether the comparison depends on them. Every variant is fitted with the same target and objective as ReSQue (Eq. 5) on the NExT-QA calibration videos, with the training settings of Supp. A.1 and A.6, and is evaluated on Qwen2.5-VL. We report the forward KL to $\boldsymbol{a}^*$ on MLVU (as in Fig. 5), the MLVU accuracy at the 6K budget, and the latency of scoring and selection in one round, measured as in Fig. 4. Both ReSQue and soft token are timed with their fused kernel implementation (Supp. A.2, *"Efficient inference with a fused kernel."*) for fair comparison.

A larger or context-aware scorer does not close the gap to ReSQue. A scorer reads each cached visual token and outputs one score per query head of the group (Supp. A.6). We vary its architecture, an MLP that sees each token alone or a two-layer transformer in which a token also attends to the mean key of each chunk of the cache, and its input, the key alone or the key and the value as in KVP [42]; all variants are fitted like ReSQue with Muon for 2,000 steps (Table 8). Neither the context of the cache nor the value helps: the transformer, with four times the parameters of the MLP, and the added value both raise the KL and lower the accuracy. Every scorer has a higher KL and a lower accuracy than ReSQue, which has fewer parameters than any of them and no per-token network to evaluate.

Scorers read the cache, since keeping hidden states would multiply its memory. Fast KVzip [29] reads the hidden states of each layer. Re-scoring the cache at every round with such a predictor requires keeping the hidden states of every cached token, which takes 2 to 3.5 times the memory of the keys and values. To compare the parameterizations under a common cache-only setting, we feed our scorers the cached keys alone or together with the values.

More soft-token embeddings stay less accurate than ReSQue. Soft tokens reach the query of each head only through the frozen model, so every round runs a forward pass of the n embeddings, which attends to the whole cache in every layer. We vary $n \in \{1, 2, 4, 8, 16, 32\}$, each initialized from the mean embeddings of the NExT-QA [70] questions plus noise and fitted with Muon for 2,000 steps (Table 7). The MLVU KL falls as n grows and is below that of ReSQue from $n = 8$ on, yet

n	Params	Train KL	MLVU KL	Acc	ms / round
1	3.6K	0.325	0.685	54.3	114
2	7.2K	0.215	0.542	54.7	116
4	14.3K	0.132	0.451	54.5	115
8	28.7K	0.072	0.369	54.3	119
16	57.3K	0.051	0.313	54.9	124
32 (default)	115K	0.063	0.315	55.4	133
ReSQue (S=8)	0.80M	0.052	0.409	55.9	93

Table 7. **Soft tokens with n embeddings** on Qwen2.5-VL. Train KL is measured on the NExT-QA calibration videos, MLVU KL and accuracy on MLVU at 6K; ms / round: round latency, measured as in Fig. 4.

Scorer	Input	Params	Train KL	MLVU KL	Acc	ms / round
MLP (default)	K	0.85M	0.067	0.414	54.7	301
MLP	K, V	1.31M	0.069	0.418	54.3	329
Transformer	K	3.5M	0.088	0.554	54.3	298
Transformer	K, V	4.0M	0.129	0.660	53.6	299
ReSQue	K	0.80M	0.052	0.409	55.9	93

Table 8. **Neural scorers** on Qwen2.5-VL, fitted with the same target and objective as ReSQue and with Muon; each scorer row is the mean of two fits with different seeds. Train KL is measured on the NExT-QA calibration videos and MLVU KL on the held-out MLVU videos; ms / round: round latency, measured as in Fig. 4; the differences among the scorers are within the measurement variation.

Query	Queries	Qwen2.5-VL	Qwen3-VL	LLaVA-OV
ReSQue (per layer and head)	LH	54.2	57.6	44.7
shared within a KV group	LH_{kv}	-1.1	-0.6	-0.1
shared within a layer	L	-7.1	-8.7	-0.1
shared across layers	H	-8.1	-9.3	-2.4
shared by the whole model	1	-9.4	-13.1	-2.4

Table 9. **Sharing the query across heads.** MLVU accuracy (%, 6K) of ReSQue with one vector per head (S=1), and the change when one query is shared by the heads of a KV group, of a layer, with the same head index across layers, or by the whole model; each variant is fitted with the same objective and data. Queries: number of distinct queries (L layers, H query heads and H_{kv} KV heads per layer).

the accuracy stays below ReSQue for every n, by 0.5 points at best ($n = 32$). Even where the soft token fits the attention better, ReSQue is more accurate for every n and faster: a round takes 93 ms against 114 to 133 ms (1.2 to 1.4×), since every n keeps the forward pass of the embeddings through the model.

D. Ablation Study of ReSQue

This section ablates the design choices of ReSQue in the query, the objective, the starting point, the calibration data and the aggregation under grouped-query attention, extending the analysis of Sec. 6.

D.1. Query

Within the query, we ablate how many vectors each head holds and how widely a query is shared across heads. All variants are fitted with the objective of Eq. 5 on the same 570 NExT-QA calibration videos.

A few vectors per head suffice: the KL saturates by $S = 8$. The query of a head holds S vectors whose maps are averaged (Eq. 8); Fig. 7b varies S from 1 to 16. The KL decreases with every added vector on all three backbones and is nearly saturated by $S = 8$. More vectors expand the representational capacity of the surrogate, and they do not collapse into one: after the fit, the $S = 8$ vectors of a head diverge (mean pairwise cosine about 0.8, against about 0.99 at the start). $S = 8$

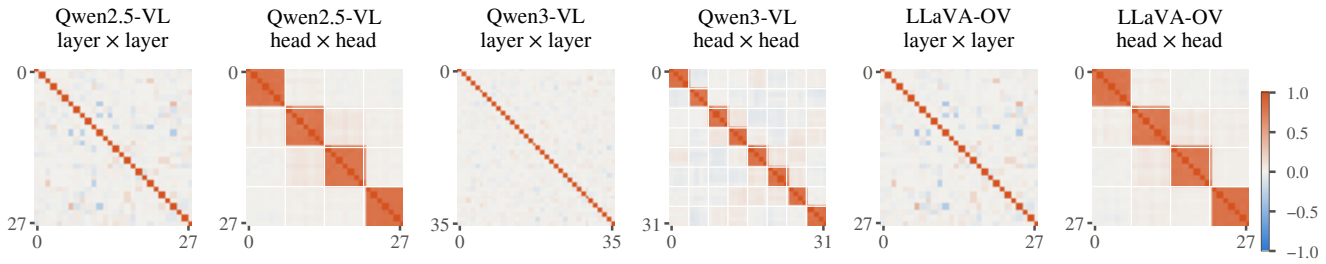


Figure 8. **Similarity of the fitted queries across layers and heads.** Cosine similarity between fitted queries (ReSQue with S=1). Layer × layer: between two layers, each represented by the queries of all its heads concatenated. Head × head: between two heads of the same layer, averaged over layers; white lines separate KV groups.

Query	Vectors	Fit	Qwen2.5-VL	Qwen3-VL	LLaVA-OV
Mean question query $\bar{\boldsymbol{q}}$	1	no	53.3	55.6	44.5
k-means centroids	8	no	53.9	55.8	44.4
Fitted, S=1	1	yes	54.2	57.6	44.7
Fitted, S=8 (ReSQue)	8	yes	**55.9**	**58.1**	**45.9**

Table 10. **Fitting versus the number of vectors.** MLVU accuracy (%, 6K) of a query without and with the fit and with one or eight vectors per head. $\bar{\boldsymbol{q}}$ is the mean of the question queries of the NExT-QA pool; the centroids are the 8 k-means centroids of the token queries of NExT-QA questions, computed for each head; the fitted rows are the S=1 and S=8 points of Fig. 7b.

improves the accuracy over $S = 1$ on every backbone, while $S = 16$ is not consistently better; we chose $S = 8$ because the KL has almost stopped decreasing by then, without claiming that it is the best value for every backbone.

Extra vectors help only when they are fitted. Table 10 asks whether the gain of $S = 8$ over $S = 1$ comes from the additional vectors or from fitting them. Without a fit, the 8 k-means centroids are within 0.6 points of the mean question query $\bar{\boldsymbol{q}}$ on all three backbones and 0.3, 1.8 and 0.3 points below the fit with $S = 1$, whereas the fit with $S = 8$ is 2.0, 2.3 and 1.5 points above the centroids. Vectors added without a fit therefore do not help; they pay off only when they are fitted to the target attention, where $S = 8$ is 1.7, 0.5 and 1.2 points above $S = 1$.

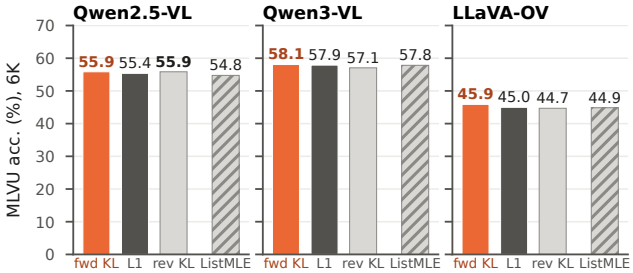
Queries must be head-specific; sharing beyond a KV group lowers the accuracy. With one vector per head ($S = 1$), where sharing across heads is well defined, we fit queries shared by a KV group, a layer, a head index across layers, or the whole model with the same objective and data (Table 9). Sharing beyond a KV group lowers the MLVU accuracy by 7 to 13 points on the Qwen models and by up to 2.4 points on LLaVA-OV, whereas sharing within a KV group changes it by about one point or less. The fitted queries are similar within a KV group but nearly orthogonal across groups on all three backbones (Fig. 8), consistent with the small effect of sharing within a group.

D.2. Objective of the Fit

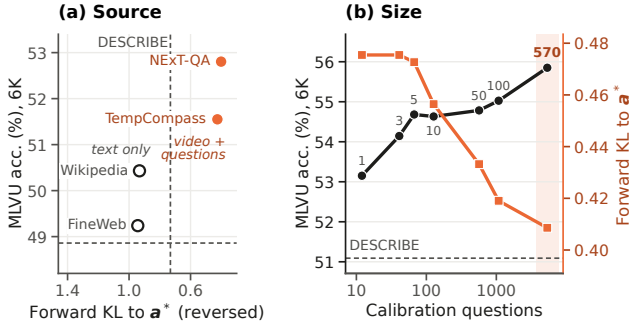
The choice of divergence matters little. The objective of Eq. 5 admits any divergence Div; we use the forward KL. Fig. 9a compares it with the L_1 distance and the reverse KL $\mathrm{KL}(\boldsymbol{a}_\theta(\mathcal{V}) \,\|\, \hat{\boldsymbol{a}}_\mathcal{V})$, and with the ranking loss ListMLE [67], which fits the order of the top 6K tokens of $\hat{\boldsymbol{a}}_\mathcal{V}$, as many as the 6K budget keeps, rather than the distribution, with $\log \boldsymbol{a}_\theta(\mathcal{V})$ as the scores; each objective is fitted in the setting of ReSQue ($S = 8$ vectors per head). The three divergences give similar accuracy, since each is minimized when the predicted attention equals the target, whose ranking gives the optimal selection of Sec. 4.1. ListMLE, which constrains only the order of the top tokens, is slightly less accurate than the forward KL on every backbone, and we adopt the forward KL as our default objective.

D.3. Initialization of the Fit

This subsection compares the starting points of the fit of Supp. A.1.



(a) **Objective of attention distribution matching.** MLVU accuracy at the 6K budget with Div in Eq. 5 the forward KL divergence, the L_1 distance, or the reverse KL divergence, or with the ranking loss ListMLE (hatched); S=8 vectors per head, fitted with Muon [28] on the 570 NExT-QA calibration videos, one fit seed. Bold: best per model.



(b) **Calibration data.** Left, source: ReSQue fitted on 100 samples of each source, a video with two questions or, for a text source, a passage with the text that follows it, averaged over the backbones. Closer to the questions is to the right. Right, size: fits on the first N NExT-QA videos on Qwen2.5-VL (N next to each point, with N=570 the ReSQue of the main experiments). Dashed: DESCRIBE.

Figure 9. **Additional ablations.** (a) Effect of the objective for attention distribution matching: divergences and a ranking loss. (b) Effect of calibration data source and size.

		before the fit		after the fit	
Backbone	Start	KL ↓	Acc ↑	KL ↓	Acc ↑
Qwen2.5-VL	$\bar{\boldsymbol{q}}$	0.81	53.3	0.41	55.9
	DESCRIBE	0.68	51.5	0.42	55.2
	$1/\sqrt{d}$	1.75	44.8	0.42	55.7
Qwen3-VL	$\bar{\boldsymbol{q}}$	0.56	55.6	0.35	58.1
	DESCRIBE	0.62	51.8	0.35	57.6
	$1/\sqrt{d}$	3.72	44.3	0.49	57.3
LLaVA-OV	$\bar{\boldsymbol{q}}$	0.83	44.5	0.34	45.9
	DESCRIBE	0.83	42.8	0.40	45.3
	$1/\sqrt{d}$	3.02	41.7	0.55	45.0

Table 11. **Starting point of the fit.** ReSQue (S=8) fitted from three starts. KL to $\boldsymbol{a}^*$ (349 MLVU videos) and MLVU accuracy (%, Acc) at the 6K budget, of the start used as the query (before the fit) and of the fitted query (after the fit).

The fit, not the starting point, determines the accuracy. We compare three starts of the fit of ReSQue (S=8): the mean question query $\bar{\boldsymbol{q}}$ (Supp. A.1), the query of DESCRIBE, and the constant vector with every element $1/\sqrt{d}$, which carries no information about the model or the video; each start is perturbed and optimized as in Supp. A.1, and the rows of $\bar{\boldsymbol{q}}$ after the fit are ReSQue of Table 13 (Table 11). Before the fit, the constant vector is far from $\boldsymbol{a}^*$ and up to 11 points less accurate than the other two starts; after the fit, the three starts are within 1 point of each other on every backbone. The objective pulls every start toward the same target attention, so the starting point matters little for the accuracy. It still matters for the KL on Qwen3-VL and LLaVA-OV, where the constant start ends farther from $\boldsymbol{a}^*$, so we keep $\bar{\boldsymbol{q}}$ as the default.

D.4. Calibration Data

The query parameters are fitted exclusively on calibration data disjoint from the evaluation benchmarks(Supp. D.1). We vary the source of the calibration data (domain and task) and then its amount.

Calibration needs videos with questions; generic text helps little. The calibration set of ReSQue pairs videos with questions, as the model meets them at inference. We fit ReSQue with the same procedure on 100 samples of each of four sources: two VideoQA sets, NExT-QA and TempCompass [35] (questions on direction, speed and order), and two public corpora of general text, FineWeb [47] and Wikipedia (Fig. 9b, left). Averaged over the three backbones, both video sources are closer to $\boldsymbol{a}^*$ than DESCRIBE (KL about 0.4 against 0.73) and more accurate, whereas the text corpora are farther from it than DESCRIBE and only slightly more accurate. Text calibration differs from deployment in two respects: it places a passage, not a video, in the context, and it fits the continuation of the passage rather than a question. Among the video

Backbone	Benchmark	6K		3K		1K	
		max	mean	max	mean	max	mean
Qwen2.5-VL	MLVU	**55.9**	55.3	**52.9**	52.3	**50.2**	48.2
	Video-MME	**59.1**	57.9	**57.2**	56.9	**54.4**	53.8
	LongVideoBench	**62.4**	61.7	60.7	**60.8**	**59.4**	58.7
	Overall	**59.1**	58.3	**56.9**	56.7	**54.7**	53.6
Qwen3-VL	MLVU	**58.1**	56.9	**55.1**	54.3	**47.8**	47.3
	Video-MME	**62.9**	62.4	60.2	**60.5**	**57.9**	57.7
	LongVideoBench	**62.6**	62.1	**61.1**	60.5	**58.6**	58.2
	Overall	**61.2**	60.5	**58.8**	58.4	**54.7**	54.4
LLaVA-OV	MLVU	**45.9**	45.1	44.0	**44.8**	**41.8**	41.6
	Video-MME	**50.2**	**50.2**	**48.9**	**48.9**	**47.9**	47.5
	LongVideoBench	**57.8**	**57.8**	**57.9**	57.2	**57.2**	55.5
	Overall	**51.3**	51.0	**50.3**	**50.3**	**49.0**	48.2

Table 12. **Aggregating the query heads of a KV group.** Accuracy (%) of ReSQue when a KV head scores each token by the maximum (default) or the mean of the scores of the query heads that share it, at about 6K / 3K / 1K visual tokens per head; the query is the same. Overall: mean over the three benchmarks. Bold: the higher of the two.

sources, TempCompass is not consistently more accurate than NExT-QA.

Ten videos already recover most of the gain. We fit ReSQue with Muon for 2,000 steps on the first N NExT-QA videos with all their questions, on Qwen2.5-VL (Fig. 9b, right). Ten videos already close 84% of the KL distance from DESCRIBE to the fit on all 570 videos; from $N = 10$ to $N = 100$ the accuracy stays nearly flat while the KL keeps falling slowly, and all 570 videos add 0.8 points. A small calibration set thus suffices to fit the query, and more videos refine it gradually.

D.5. Aggregation under Grouped-Query Attention

Max over query heads is slightly better than mean. The query heads that share a KV head each give a score to every cached token, and the KV head keeps the B tokens with the largest maximum of these scores (Supp. A.2). Replacing the maximum by the mean over the query heads, with the same query and protocol (Table 12), lowers the accuracy in 21 of the 27 settings of Table 13, but by at most 1.1 points averaged over the three benchmarks. The maximum is an empirical design choice: the expected attention retention of Sec. 4.1, summed over the query heads of a group, would instead call for their sum or mean. We use maximum aggregation as a fixed default across all backbones and budgets.

E. Limitations and Future Work

This section states the limitations of ReSQue and the directions we leave for future work.

Limitations. First, ReSQue prunes tokens and does nothing else to the cache. Methods that merge evicted tokens into the kept ones [58, 79] or quantize the cache [24, 38] are orthogonal to it, and combining them with ReSQue may give better accuracy at the same memory; we have not tested such a combination. Second, ReSQue scores a token by its attention weight alone, which does not reflect what the token contributes to the output. The contribution of a token to the output of a head is its attention weight multiplied by its value vector, so two tokens with the same weight are ranked alike whatever their values. Value-aware scores [20] include the value by weighting the attention with the norm of the value vector.

Future work. We outline two directions. First, ReSQue gives every head the same budget, so that the queries we compare differ only in the tokens they select. Methods that allocate the budget across heads, by the concentration of their attention [14] or to heads identified offline as retrieval or visual heads [45, 61, 69], are orthogonal to the choice of query and could be applied on top of ReSQue. Second, a value term such as the value norm of VATP [20] can be added to the score of ReSQue; we have not tested whether it improves the selection.

Method	MLVU	Video-MME	LongVideoBench	Overall
Qwen2.5-VL-7B				
Full KV	54.0	59.4	63.9	59.1
SWA	45.9/39.9/40.8	50.9/48.8/47.0	60.0/57.6/57.3	52.3/48.8/48.4
InfiniPot-V	50.3/47.7/43.4	56.9/54.6/52.3	62.0/60.3/58.2	56.4/54.2/51.3
CoRDS	48.6/47.4/42.4	54.6/52.8/49.4	61.9/59.2/57.9	55.0/53.1/49.9
KeyDiff	52.3/48.6/42.4	55.3/52.8/50.8	62.0/**60.7**/58.4	56.6/54.0/50.5
StreamMem	49.7/45.2/44.3	56.5/54.2/49.8	60.3/58.0/58.3	55.5/52.5/50.8
HERMES	53.0/49.7/42.9	57.2/55.1/51.8	60.6/59.5/58.7	56.9/54.7/51.1
Expected Attention	48.4/48.9/44.9	56.6/54.5/51.6	61.9/59.5/58.0	55.6/54.3/51.5
TriAttention	53.2/50.8/49.4	58.2/55.8/53.7	62.3/60.4/59.3	57.9/55.6/54.2
ReSQue	**55.9/52.9/50.2**	**59.1/57.2/54.4**	**62.4/60.7/59.4**	**59.1/56.9/54.7**
Qwen3-VL-8B				
Full KV	62.4	65.2	64.5	64.0
SWA	41.4/39.9/38.1	52.4/50.3/47.5	56.5/54.8/52.0	50.1/48.4/45.9
InfiniPot-V	54.6/48.0/43.1	61.4/58.7/53.1	60.9/58.7/54.8	58.9/55.1/50.4
CoRDS	51.9/50.1/43.1	57.8/54.3/49.6	60.5/58.7/53.1	56.7/54.4/48.6
KeyDiff	48.2/45.5/39.3	54.9/53.2/47.6	58.8/55.5/49.6	54.0/51.4/45.5
StreamMem	50.0/46.0/39.7	59.1/55.0/49.1	59.6/57.3/54.3	56.3/52.8/47.7
HERMES	48.6/44.1/41.2	57.6/55.3/50.1	59.1/58.0/55.5	55.1/52.5/48.9
Expected Attention	50.8/46.1/43.2	60.7/57.5/53.9	60.9/58.3/54.9	57.5/54.0/50.7
TriAttention	54.2/51.0/46.3	62.0/58.1/53.6	60.7/59.3/55.1	59.0/56.1/51.7
ReSQue	**58.1/55.1/47.8**	**62.9/60.2/57.9**	**62.6/61.1/58.6**	**61.2/58.8/54.7**
LLaVA-OV-7B				
Full KV	47.6	50.6	58.8	52.3
SWA	40.3/38.6/36.7	48.2/46.0/45.0	57.3/56.5/56.0	48.6/47.0/45.9
InfiniPot-V	42.7/40.3/38.6	49.4/47.9/46.7	**58.2**/57.2/**57.2**	50.1/48.5/47.5
CoRDS	42.8/40.8/37.3	49.5/47.1/45.2	57.6/56.7/55.5	50.0/48.2/46.0
KeyDiff	44.8/42.6/40.0	48.1/47.7/46.1	**58.2**/57.6/55.7	50.4/49.3/47.3
StreamMem	43.1/42.3/38.5	48.8/48.1/47.2	57.3/56.9/55.8	49.7/49.1/47.2
HERMES	41.0/39.3/37.0	48.3/47.3/44.9	57.1/56.9/56.2	48.8/47.8/46.1
Expected Attention	43.6/41.7/38.5	49.1/47.7/46.5	57.9/57.0/54.9	50.2/48.8/46.6
TriAttention	45.2/**44.8**/41.6	49.4/48.0/46.8	57.9/57.0/55.7	50.8/49.9/48.0
ReSQue	**45.9**/44.0/**41.8**	**50.2/48.9/47.9**	57.8/**57.9/57.2**	**51.3/50.3/49.0**

Table 13. **Long-video question answering under streaming compression.** Each cell: accuracy (%) at budgets of about 6K / 3K / 1K visual tokens per head. MLVU: average over task groups; Video-MME: average over medium and long videos; LongVideoBench: average over the four video-length groups of the validation set; Overall: mean of the three benchmarks. **Bold**: best, underline: second best per budget, column and backbone, Full KV excluded. ReSQue: fitted query $\boldsymbol{q}^\star$ with eight vectors per head.

F. Full Results

This section gives the numbers behind the figures of the main paper: the accuracy of every compressor on the long-video benchmarks (Supp. F.1), its breakdown by task group, video duration and video length and the streaming benchmark (Supp. F.2), and the accuracy of every query of the query analysis at the three budgets (Supp. F.3).

F.1. Main Results

Table 13 gives the accuracy of every compressor on MLVU, Video-MME and LongVideoBench at the 6K, 3K and 1K budgets on the three backbones, with Full KV as reference: the numbers from which the comparison of Fig. 2 is drawn.

Method	MLVU				Video-MME			LongVideoBench			
	Single	Multi	Holistic	Avg	Medium	Long	Avg	≤60 s	600 s	3600 s	Avg
Qwen2.5-VL-7B											
Full KV	48.0	38.1	75.9	54.0	64.7	54.2	59.4	69.8	63.6	52.3	63.9
SWA	36.3	35.4	66.0	45.9	52.4	49.3	50.9	70.9	50.5	47.7	60.0
InfiniPot-V	45.6	33.6	71.6	50.3	60.9	53.0	56.9	70.6	58.5	48.0	62.0
CoRDS	41.1	33.6	70.9	48.6	57.6	51.6	54.6	71.2	57.3	47.9	61.9
KeyDiff	42.3	38.1	**76.6**	52.3	58.3	52.3	55.3	71.2	55.1	**50.5**	62.0
StreamMem	42.3	33.6	73.0	49.7	60.0	53.0	56.5	71.2	53.2	45.7	60.3
HERMES	44.0	**43.4**	71.6	53.0	58.9	55.4	57.2	70.9	52.9	47.7	60.6
Expected Attention	44.0	31.9	69.5	48.4	59.8	53.3	56.6	**71.5**	57.0	47.5	61.9
TriAttention	47.6	38.9	73.0	53.2	**62.6**	53.8	58.2	**71.5**	56.8	49.3	62.3
ReSQue	**49.2**	42.5	75.9	**55.9**	61.7	**56.4**	**59.1**	70.6	**59.5**	48.6	**62.4**
Qwen3-VL-8B											
Full KV	55.6	51.3	80.1	62.4	69.2	61.1	65.2	67.6	66.0	56.9	64.5
SWA	32.3	31.0	61.0	41.4	53.0	51.8	52.4	69.0	45.9	42.4	56.5
InfiniPot-V	50.0	**44.2**	69.5	54.6	65.7	57.1	61.4	68.7	59.0	47.3	60.9
CoRDS	47.2	39.8	68.8	51.9	61.8	53.9	57.8	67.3	60.9	46.5	60.5
KeyDiff	43.1	35.4	66.0	48.2	57.3	52.6	54.9	68.7	53.2	45.0	58.8
StreamMem	45.6	33.6	70.9	50.0	62.3	55.9	59.1	69.0	55.6	45.4	59.6
HERMES	39.9	39.8	66.0	48.6	58.9	56.3	57.6	67.3	55.3	46.5	59.1
Expected Attention	45.2	36.3	70.9	50.8	64.2	57.1	60.7	**69.3**	58.7	46.6	60.9
TriAttention	50.0	38.9	73.8	54.2	65.0	59.0	62.0	67.0	61.9	47.0	60.7
ReSQue	**52.8**	43.4	**78.0**	**58.1**	**65.9**	**60.0**	**62.9**	**69.3**	**62.9**	**49.1**	**62.6**
LLaVA-OV-7B											
Full KV	43.1	23.0	76.6	47.6	54.7	46.6	50.6	66.2	53.4	49.3	58.8
SWA	33.5	22.1	65.2	40.3	51.7	44.8	48.2	66.5	49.8	46.5	57.3
InfiniPot-V	35.1	22.1	70.9	42.7	51.6	**47.3**	49.4	66.5	53.4	46.3	**58.2**
CoRDS	36.3	**24.8**	67.4	42.8	52.2	46.8	49.5	66.5	52.2	45.2	57.6
KeyDiff	38.3	23.9	72.3	44.8	50.7	45.4	48.1	**66.8**	**54.1**	45.2	**58.2**
StreamMem	37.1	21.2	70.9	43.1	52.6	45.0	48.8	66.5	51.2	45.0	57.3
HERMES	33.1	**24.8**	65.2	41.0	51.2	45.3	48.3	66.5	50.5	45.0	57.1
Expected Attention	35.1	**24.8**	70.9	43.6	52.3	45.8	49.1	**66.8**	53.2	44.9	57.9
TriAttention	**39.5**	23.0	73.0	45.2	52.6	46.2	49.4	66.5	52.9	45.7	57.9
ReSQue	39.1	**24.8**	**73.8**	**45.9**	**54.0**	46.4	**50.2**	65.9	52.7	**46.6**	57.8

Table 14. **Task-level accuracy at the 6K budget.** The numbers of Table 13 split by task group (MLVU), video duration (Video-MME medium and long videos) and video length (LongVideoBench: videos of at most 60, 600 and 3600 seconds; the 15 and 60 second groups are merged), in %; Avg is the value of the main table, the macro average (unweighted mean) of the groups of each benchmark, which for LongVideoBench are the four length groups of 15, 60, 600 and 3600 seconds. Bold: best, underlined: second best per column and backbone.

F.2. Per-Category Results

Long-video benchmarks. Tables 14, 15 and 16 split the accuracies of Table 13 by task group (MLVU), video duration (Video-MME medium and long videos) and video length (LongVideoBench), at each budget.

	MLVU				Video-MME			LongVideoBench			
Method	Single	Multi	Holistic	Avg	Medium	Long	Avg	≤60 s	600 s	3600 s	Avg
Qwen2.5-VL-7B											
Full KV	48.0	38.1	75.9	54.0	64.7	54.2	59.4	69.8	63.6	52.3	63.9
SWA	30.2	25.7	63.8	39.9	50.0	47.6	48.8	68.1	48.8	45.7	57.6
InfiniPot-V	40.7	33.6	68.8	47.7	58.7	50.6	54.6	70.6	53.6	46.3	60.3
CoRDS	39.5	31.9	70.9	47.4	55.8	49.8	52.8	69.5	52.2	45.6	59.2
KeyDiff	37.1	32.7	**75.9**	48.6	56.1	49.4	52.8	69.8	53.9	**49.5**	**60.7**
StreamMem	35.9	31.0	68.8	45.2	56.1	52.3	54.2	68.7	51.2	43.6	58.0
HERMES	37.5	**39.8**	71.6	49.7	55.8	**54.3**	55.1	70.4	50.0	47.2	59.5
Expected Attention	42.7	34.5	69.5	48.9	56.9	52.1	54.5	**70.9**	51.2	45.0	59.5
TriAttention	42.7	37.2	72.3	50.8	59.6	52.0	55.8	69.5	55.8	46.8	60.4
ReSQue	**44.4**	**39.8**	74.5	**52.9**	**60.8**	53.7	**57.2**	69.0	**57.8**	47.2	**60.7**
Qwen3-VL-8B											
Full KV	55.6	51.3	80.1	62.4	69.2	61.1	65.2	67.6	66.0	56.9	64.5
SWA	29.0	31.9	58.9	39.9	50.0	50.7	50.3	66.8	45.6	40.6	54.8
InfiniPot-V	44.0	35.4	64.5	48.0	61.3	56.1	58.7	**68.7**	53.9	43.8	58.7
CoRDS	46.4	40.7	63.1	50.1	56.8	51.9	54.3	67.3	56.3	43.8	58.7
KeyDiff	36.3	36.3	63.8	45.5	54.0	52.3	53.2	68.1	46.6	39.4	55.5
StreamMem	39.5	35.4	63.1	46.0	55.4	54.6	55.0	67.3	53.2	41.7	57.3
HERMES	35.9	31.9	64.5	44.1	56.3	54.2	55.3	67.6	52.9	44.0	58.0
Expected Attention	38.3	32.7	67.4	46.1	58.9	56.1	57.5	67.0	55.3	43.8	58.3
TriAttention	46.4	37.2	69.5	51.0	60.7	55.4	58.1	67.3	59.0	44.0	59.3
ReSQue	**49.2**	**41.6**	**74.5**	**55.1**	**62.9**	**57.4**	**60.2**	67.9	**61.9**	**47.2**	**61.1**
LLaVA-OV-7B											
Full KV	43.1	23.0	76.6	47.6	54.7	46.6	50.6	66.2	53.4	49.3	58.8
SWA	33.5	21.2	61.0	38.6	49.0	43.0	46.0	67.0	47.6	44.3	56.5
InfiniPot-V	33.1	21.2	66.7	40.3	50.1	45.8	47.9	65.9	51.7	45.4	57.2
CoRDS	34.3	23.0	65.2	40.8	50.0	44.2	47.1	66.2	49.5	44.7	56.7
KeyDiff	35.1	23.9	68.8	42.6	50.2	45.2	47.7	67.0	51.0	**45.6**	57.6
StreamMem	33.5	24.8	68.8	42.3	**51.9**	44.3	48.1	**67.3**	48.5	44.3	56.9
HERMES	35.1	20.4	62.4	39.3	49.2	45.4	47.3	67.0	49.0	44.3	56.9
Expected Attention	31.9	**26.5**	66.7	41.7	50.7	44.8	47.7	66.5	50.7	44.3	57.0
TriAttention	**40.3**	23.9	70.2	**44.8**	51.2	44.8	48.0	65.9	51.2	45.0	57.0
ReSQue	37.1	23.9	**70.9**	44.0	51.8	**46.0**	**48.9**	66.5	**52.9**	**45.6**	**57.9**

Table 15. **Task-level accuracy at the 3K budget.** The numbers of Table 13 split by task group (MLVU), video duration (Video-MME medium and long videos) and video length (LongVideoBench: videos of at most 60, 600 and 3600 seconds; the 15 and 60 second groups are merged), in %; Avg is the value of the main table, the macro average (unweighted mean) of the groups of each benchmark, which for LongVideoBench are the four length groups of 15, 60, 600 and 3600 seconds. Bold: best, underlined: second best per column and backbone.

	MLVU				Video-MME			LongVideoBench			
Method	Single	Multi	Holistic	Avg	Medium	Long	Avg	≤60 s	600 s	3600 s	Avg
Qwen2.5-VL-7B											
Full KV	48.0	38.1	75.9	54.0	64.7	54.2	59.4	69.8	63.6	52.3	63.9
SWA	29.8	30.1	62.4	40.8	48.0	46.0	47.0	69.5	47.1	43.3	57.3
InfiniPot-V	34.3	29.2	66.7	43.4	55.1	49.4	52.3	69.5	49.8	44.3	58.2
CoRDS	28.6	31.9	66.7	42.4	51.3	47.4	49.4	69.8	49.0	43.3	57.9
KeyDiff	27.0	32.7	67.4	42.4	53.2	48.4	50.8	69.3	50.2	45.2	58.4
StreamMem	33.1	31.0	68.8	44.3	51.2	48.4	49.8	70.6	48.8	43.4	58.3
HERMES	34.7	27.4	66.7	42.9	53.7	49.9	51.8	**70.9**	46.6	**46.5**	58.7
Expected Attention	35.1	30.1	69.5	44.9	53.2	50.0	51.6	69.3	47.6	46.3	58.0
TriAttention	37.1	**38.1**	**73.0**	49.4	56.3	51.1	53.7	69.0	**53.2**	**46.5**	59.3
ReSQue	**39.5**	**38.1**	**73.0**	**50.2**	**56.9**	**51.9**	**54.4**	69.3	52.9	**46.5**	**59.4**
Qwen3-VL-8B											
Full KV	55.6	51.3	80.1	62.4	69.2	61.1	65.2	67.6	66.0	56.9	64.5
SWA	25.4	32.7	56.0	38.1	48.0	47.0	47.5	64.5	42.5	36.9	52.0
InfiniPot-V	33.9	34.5	61.0	43.1	54.6	51.7	53.1	**66.8**	45.4	40.8	54.8
CoRDS	36.3	29.2	63.8	43.1	50.2	48.9	49.6	65.9	43.7	37.4	53.1
KeyDiff	27.8	32.7	57.4	39.3	48.3	46.9	47.6	62.3	39.6	35.1	49.6
StreamMem	31.5	31.0	56.7	39.7	47.8	50.3	49.1	63.4	49.8	41.3	54.3
HERMES	33.5	32.7	57.4	41.2	49.6	50.6	50.1	66.2	47.8	42.0	55.5
Expected Attention	33.9	32.7	63.1	43.2	54.7	53.2	53.9	65.1	48.8	41.0	54.9
TriAttention	35.1	**37.2**	**66.7**	46.3	54.3	52.9	53.6	64.3	52.7	39.7	55.1
ReSQue	**40.3**	36.3	**66.7**	**47.8**	**59.9**	**55.9**	**57.9**	66.2	**59.2**	**42.9**	**58.6**
LLaVA-OV-7B											
Full KV	43.1	23.0	76.6	47.6	54.7	46.6	50.6	66.2	53.4	49.3	58.8
SWA	30.6	18.6	61.0	36.7	47.0	43.0	45.0	66.5	49.0	42.2	56.0
InfiniPot-V	31.9	19.5	64.5	38.6	48.2	45.1	46.7	**67.9**	49.5	43.3	**57.2**
CoRDS	31.5	19.5	61.0	37.3	47.4	42.9	45.2	65.9	46.8	43.3	55.5
KeyDiff	31.9	22.1	66.0	40.0	48.1	44.1	46.1	66.8	46.1	43.1	55.7
StreamMem	30.2	22.1	63.1	38.5	**51.4**	43.0	47.2	67.6	45.6	42.4	55.8
HERMES	31.0	20.4	59.6	37.0	46.9	43.0	44.9	66.2	48.8	43.8	56.2
Expected Attention	34.3	20.4	61.0	38.5	48.7	44.3	46.5	65.7	46.6	41.5	54.9
TriAttention	34.3	23.9	**66.7**	41.6	50.2	43.3	46.8	65.4	48.3	43.8	55.7
ReSQue	**34.7**	**24.8**	66.0	**41.8**	50.7	**45.2**	**47.9**	65.4	**52.9**	**45.2**	**57.2**

Table 16. **Task-level accuracy at the 1K budget.** The numbers of Table 13 split by task group (MLVU), video duration (Video-MME medium and long videos) and video length (LongVideoBench: videos of at most 60, 600 and 3600 seconds; the 15 and 60 second groups are merged), in %; Avg is the value of the main table, the macro average (unweighted mean) of the groups of each benchmark, which for LongVideoBench are the four length groups of 15, 60, 600 and 3600 seconds. Bold: best, underlined: second best per column and backbone.

Method	Scene sum.	Action cap.	Event order	Order (Y/N)	Happened (Y/N)	All
Qwen2.5-VL-7B						
SWA	59.4	49.8	33.2	48.2	70.9	52.9
InfiniPot-V	59.4	53.2	33.1	49.7	72.1	54.0
CoRDS	59.3	54.8	**33.9**	49.2	**72.5**	54.4
KeyDiff	59.3	54.0	29.8	51.1	70.5	53.5
StreamMem	58.3	54.2	28.3	48.8	68.8	52.2
HERMES	59.8	55.7	29.8	48.8	69.1	53.1
Expected Attention	59.8	54.5	32.4	50.4	68.1	53.6
TriAttention	59.0	54.2	28.5	**52.1**	68.5	53.1
ReSQue	**62.1**	**59.2**	30.8	49.7	70.1	**54.9**
Qwen3-VL-8B						
SWA	62.5	58.8	**35.2**	61.7	70.6	58.4
InfiniPot-V	65.9	57.3	32.1	61.9	72.1	58.6
CoRDS	65.5	59.0	31.6	62.8	72.3	59.0
KeyDiff	63.8	56.0	29.8	58.7	**73.2**	57.1
StreamMem	64.2	58.5	34.2	62.2	69.8	58.4
HERMES	66.1	60.7	34.4	60.8	70.5	59.1
Expected Attention	64.0	58.0	34.5	61.0	72.5	58.7
TriAttention	66.8	60.8	34.0	**63.5**	70.1	**59.7**
ReSQue	**68.5**	**63.5**	31.9	61.6	69.6	**59.7**
LLaVA-OV-7B						
SWA	50.4	41.2	16.4	66.7	69.1	49.9
InfiniPot-V	48.5	41.0	17.1	65.6	69.2	49.4
CoRDS	48.2	42.8	16.4	65.7	**71.3**	50.0
KeyDiff	48.6	42.5	17.6	65.1	69.2	49.7
StreamMem	49.2	43.3	16.9	65.5	69.9	50.1
HERMES	49.3	40.7	18.2	65.5	69.6	49.8
Expected Attention	47.7	40.0	16.9	**67.0**	70.2	49.5
TriAttention	49.0	41.3	16.9	66.2	67.6	49.3
ReSQue	**55.0**	**48.8**	**18.9**	**67.0**	68.2	**52.6**

Table 17. **VStream-QA accuracy by question type.** The runs of Table 1 (about 6K visual tokens per head), split by the type of the question, over the ego-centric (RVS-Ego) and movie (RVS-Movie) splits together; the number of questions is 734 (Scene sum.), 600 (Action cap.), 614 (Event order), 724 (Order (Y/N)), 698 (Happened (Y/N)). Scene sum.: scene summary, Action cap.: action caption, Event order: what is the order of the events, Order (Y/N): judging whether an order is correct, Happened (Y/N): whether something happened. Open-ended answers are judged by `gpt-5-mini`. **Bold**: best, underline: second best per column and backbone.

Streaming benchmark. Table 17 splits the accuracies of Table 1 by question type, over the ego-centric (RVS-Ego) and movie (RVS-Movie) splits together.

F.3. Query Analysis Across Budgets

The query analysis of Sec. 6 compares the queries at the 6K budget; Table 18 gives the MLVU accuracy of every one of them at the 6K, 3K and 1K budgets, with the selector and the streaming protocol of Supp. A.3 held fixed (the queries are described in Supp. A.5). Expected Attention is the query-only version of Supp. A.5; the TriAttention row is copied from Table 13, with its full score. Q-Filters, which induces no attention distribution and is therefore not among the queries of Fig. 5 (Supp. A.5), is less accurate than ReSQue in all nine settings.

	Qwen2.5-VL			Qwen3-VL			LLaVA-OV		
Query	6K	3K	1K	6K	3K	1K	6K	3K	1K
Recent visual tokens	49.4	47.0	44.8	49.2	45.5	43.5	43.6	42.6	38.5
Q-Filters (vision)	49.2	48.3	44.4	54.5	52.2	44.6	44.1	41.4	39.6
Chat template	49.0	43.8	44.3	53.5	47.7	40.7	42.6	39.8	38.2
DESCRIBE	51.1	47.7	43.9	53.3	48.7	42.7	42.2	40.7	37.0
EXPLAIN	49.4	44.6	43.5	52.5	45.7	38.8	41.6	42.3	37.2
SUMMARIZE	49.6	46.8	45.9	50.8	46.6	41.8	42.8	41.2	39.2
Expected Attention	50.3	47.9	45.1	52.6	48.1	42.2	42.5	42.3	38.8
TriAttention	53.2	50.8	49.4	54.2	51.0	46.3	45.2	**44.8**	41.6
ReSQue	**55.9**	**52.9**	**50.2**	**58.1**	**55.1**	**47.8**	**45.9**	44.0	**41.8**

Table 18. **Accuracy of each query at the three budgets.** MLVU accuracy (%, average over task groups) when the cache is compressed with each query, at about 6K / 3K / 1K visual tokens per head, with the selector, the streaming protocol and the budgets of Supp. A.3 held fixed. Expected Attention uses the mean and covariance of the visual-token queries of the newest chunk, without the value norm, so that only the query differs; TriAttention is the row of Table 13, with its full score. Bold: best, underlined: second best per column.



Figure 10. **Qualitative example** (MLVU, LLaVA-OV, 6K tokens per head). Patches each method keeps in one head (layer 16, KV head 0) when the question arrives, with evicted patches faded and the share kept under each frame. The blue frames lie in the evidence clip (7:45–8:07), which receives 51% of the question's attention on the uncompressed cache but spans only 5.0% of the frames. ReSQue keeps 30% of the patches of the evidence frames, against 10% and 8%; all three methods answer correctly.

G. Additional Qualitative Results

Fig. 10 to Fig. 12 show three more questions in the format of Fig. 3: one from MLVU, whose evidence clips are labelled, and two from LongVideoBench, whose evidence is the annotated moment the question refers to. In each, we draw the head whose question attention on the uncompressed cache is the most concentrated on the evidence, and compare what InfiniPot-V, TriAttention and ReSQue hold in it when the question arrives (6K tokens per head).

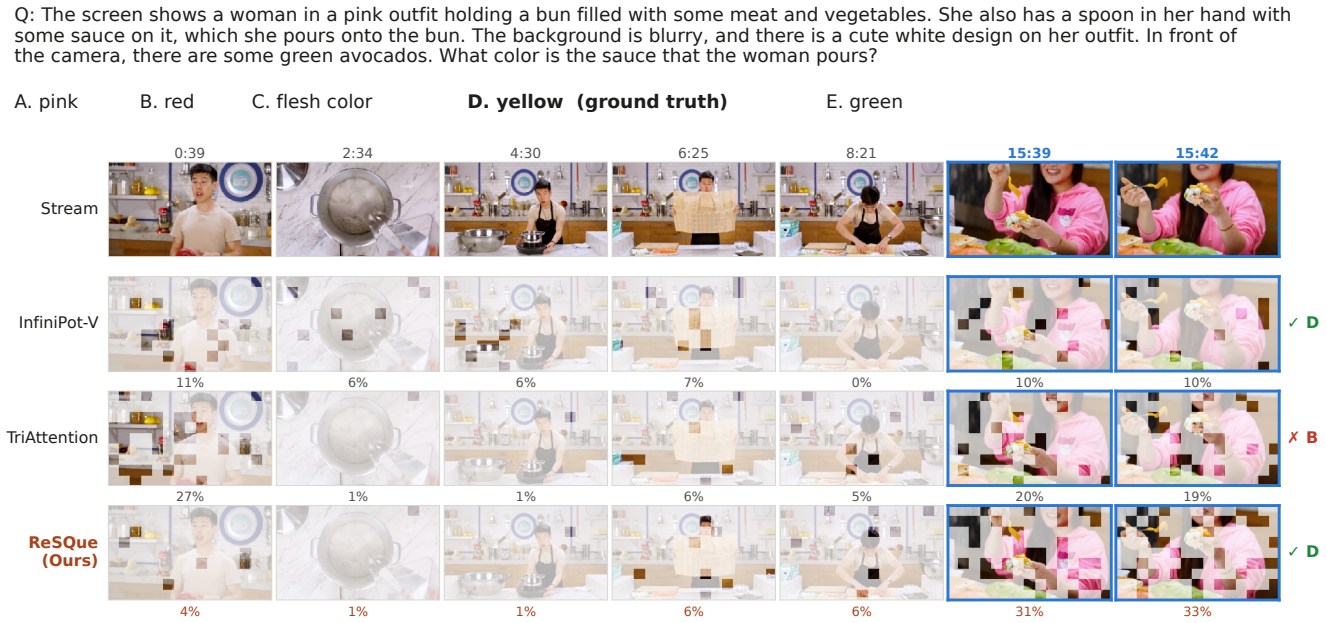


Figure 11. **Qualitative example** (LongVideoBench, Qwen3-VL, 6K tokens per head). Patches each method keeps in one head (layer 23, KV head 2) when the question arrives, with evicted patches faded and the share kept under each frame. The blue frames lie at the moment the question refers to (about 15:40), which receives 14% of the question's attention on the uncompressed cache but spans only 0.5% of the frames. ReSQue keeps 32% of their patches, against 10% for InfiniPot-V and 19% for TriAttention; InfiniPot-V and ReSQue answer correctly, TriAttention does not.

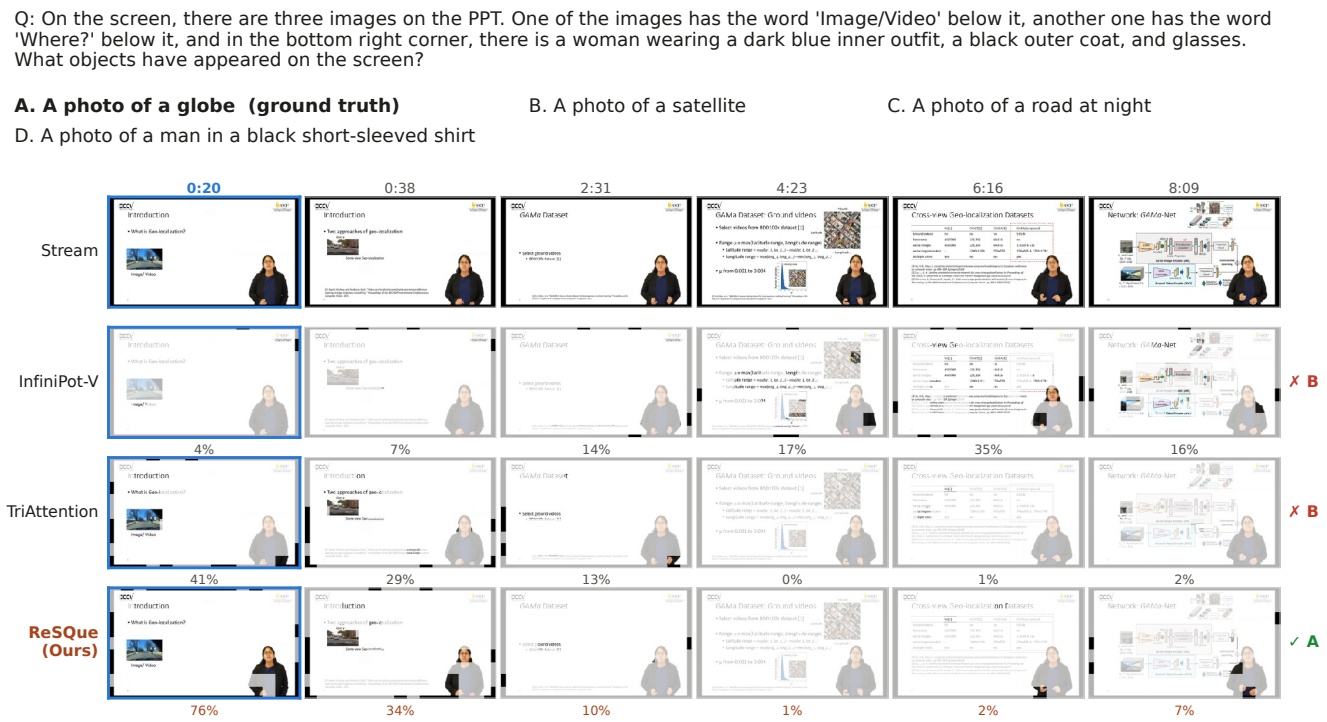


Figure 12. **Qualitative example** (LongVideoBench, Qwen2.5-VL, 6K tokens per head). Patches each method keeps in one head (layer 14, KV head 0) when the question arrives, with evicted patches faded and the share kept under each frame. The blue frame is the moment the question refers to (about 0:20), which receives 6% of the question's attention on the uncompressed cache but spans only 0.3% of the frames. ReSQue keeps 76% of its patches in this head, against 4% and 41% (37% averaged over all heads, against 9% and 15%), and is the only one to answer correctly.