Going Down Memory Lane: Scaling Tokens for Video Stream Understanding with Dynamic KV-Cache Memory

Vatsal Agarwal*
University of Maryland, College Park
vatsalag@umd.edu

Saksham Suri
University of Maryland, College Park

Matthew Gwilliam
TikTok

Pulkit Kumar
University of Maryland, College Park

Abhinav Shrivastava
University of Maryland, College Park

Abstract

Online video understanding requires models to robustly encode, store, and retrieve information from a continuous stream to support accurate question answering. Existing approaches rely on key–value caching to accumulate frame-level representations over time, but allocate only a limited token budget per frame, causing loss of fine-grained visual detail. We find that naively scaling this budget harms retrieval quality and question-answering performance, as denser token streams introduce redundancy that degrades query–frame similarity. To address this, we first introduce an adaptive selection strategy that reduces token redundancy while preserving local spatiotemporal information. Second, we propose a training-free retrieval ensemble that leverages external models to better identify relevant frames. Our method, **MemStream**, achieves +8.0% on CG-Bench, +8.5% on LVBench, and +2.4% on VideoMME (Long) over ReKV with Qwen2.5-VL-7B. Furthermore, we show that MemStream consistently improves performance across several Video-LLMs for long-video understanding. Project page: https://vatsalag99.github.io/memstream/

1 Introduction

Video understanding is a complex task requiring a model to perceive and reason about the scene, the objects in it, and temporal interactions such as actions or events. Recent developments in multimodal large language models (MLLMs) have equipped them with the capabilities to understand these complex relations (1; 2; 3; 4; 5; 6; 7; 8; 9). However, processing long videos remains a significant challenge due to limited context length. Capturing fine-grained visual details requires the model to encode and retain information at both high spatial and temporal granularity, yet current models must sacrifice this ability to fit within the context budget.

Existing approaches use a combination of temporal subsampling, where a subset of key frames is selected from the video (10; 11; 12), or spatial subsampling (9; 13; 4), where a subset of tokens are processed for each frame. However, sampling only a sparse subset of frames results in the

*Work partially done during internship at TikTok. This work is for research purposes only and is not currently integrated into any TikTok technology.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

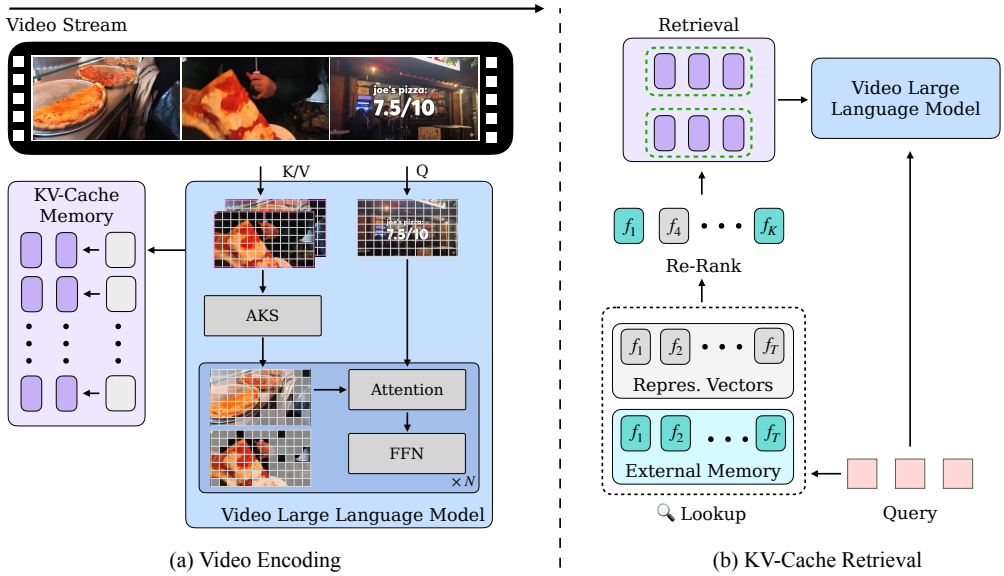


Figure 1: (a) We propose constructing the key–value cache with sparse sliding-window attention using our adaptive key selection (AKS) strategy. (b) During question-answering, we merge complementary retrieval signals from external models via a training-free ensemble.

model missing granular temporal information such as short but critical events (14; 15), while low frame-wise token budgets can result in the model missing fine-grained visual details (16). Recent dynamic-resolution MLLMs such as Qwen2.5-VL (13) make it possible to process frames at higher spatial token density; however, doing so across long videos still causes it to exceed the context budget.

A natural way to address these limitations is by processing the video in a streaming manner, where frames are processed as they arrive and stored incrementally for question-answering. This is exemplified by ReKV (14), which encodes video via causal sliding-window attention and stores the resulting representations directly in the model's key–value (KV) cache. During question-answering, relevant frames are retrieved from this cache across each layer. In this way, the model no longer needs to re-encode the video each time for a new question and can process the video beyond its original context length. Moreover, the KV-cache can serve as an efficient memory that can be queried to answer multiple questions. Crucially, this design couples encoding, retrieval, and question-answering through a single set of KV representations, making model performance fundamentally dependent on their quality. If these cache representations are not discriminative or fail to capture fine-grained visual information, the model may fail to retrieve the right visual evidence or produce incorrect answers.

To better understand this bottleneck, we conduct a detailed analysis of existing KV-cache-based methods. Our exploration reveals two critical limitations of this design. First, we observe that increasing per-frame token budget counterintuitively degrades both retrieval quality and downstream question-answering performance. We identify that this is a result of the sliding-window attention, which we hypothesize amplifies local redundancy within the key–value features, resulting in less discriminative frame-wise representations. Second, we find that retrieval quality is unreliable even at lower token budgets: while some layers consistently identify the relevant video segment, others miss it entirely. Moreover, we observe that internal KV features alone often lack sufficient fine-grained visual detail, particularly for questions involving precise object attributes or subtle motion cues.

Motivated by these findings, we propose MemStream, a training-free framework for online video understanding that addresses both failure modes. First, we introduce Adaptive Key Selection (AKS) during video encoding to preserve informative content in the sliding window while discarding redundant signals. In this way, AKS produces discriminative KV-cache representations that enhance retrieval fidelity and improve downstream question-answering performance. To address variable retrieval quality, we propose a training-free retrieval ensemble that utilizes external models to

supplement internal KV features. This design leverages complementary retrieval signals across internal and external features, yielding more consistent retrieval across layers and improved overall retrieval quality. Furthermore, our retrieval ensemble enables the integration of stronger vision encoders as they become available, without any retraining.

In summary, our core contributions are: **(1)** an extensive analysis revealing that KV-cache-based methods suffer from representational degradation at high token budgets and unreliable retrieval quality; **(2)** Adaptive Key Selection (AKS), a sparse sliding-window attention strategy that reduces redundancy while preserving informative spatiotemporal content; and **(3)** a training-free retrieval ensemble that combines internal KV-cache retrieval with external vision-language retrieval signals.

2 Related Work

Long Video Understanding Recent advances have endowed Video-LLMs with powerful video understanding capabilities (3; 6; 5; 13; 2). These models typically consist of a vision encoder to process frames (17; 18; 19), and a large-language model (LLM) (20; 21; 3) to generate text responses based on user input (1; 13). Despite this progress, these models still struggle with long-form video due to limited context length, making precise temporal localization and reasoning particularly challenging (22; 23; 24). To address these issues, existing approaches have primarily focused on designing effective compression strategies via spatial and temporal subsampling (25; 26; 27; 28; 29; 30; 31; 24). Recently, (32) proposed a training-based approach to introduce learned sparsity to existing Video-LLMs, showcasing improved performance on long-form video understanding. While token-compression methods mitigate redundancy, they risk losing critical fine-grained information. In contrast, our approach selectively preserves tokens only within the local attention window, maintaining a non-evicting KV-cache memory for long-term storage in a training-free manner.

Streaming Video Understanding Streaming video understanding requires the model to process and store frames as they arrive in an online manner. Then, during question-answering, the stored features are queried across a variety of different types of questions (14; 33; 34; 8; 35). Recent research has explored many different ways to process and store this visual information to obtain an optimal trade-off between efficiency and accuracy. One key area of interest is designing an effective memory for storing and retrieving frame information (36; 37; 38; 14; 39). These methods broadly either store and compress visual information before feeding it to the LLM (40; 37; 36), or use the LLM's internal KV-cache to store intermediate visual features (39; 14; 41; 38; 42). Several works have also explored compression strategies for streaming video (39; 31; 38; 42; 43). These methods either apply compression on tokens before feeding them into the Video-LLM (31; 38), or apply compression directly on the KV-cache (42; 39). While these strategies improve efficiency, they discard discrete frame-level information. This not only loses per-frame detail but also reduces retrieval expressivity: different questions may require different subsets of frames, yet a compressed cache must serve all queries with the same fixed representation, collapsing an exponentially large retrieval space into one.

3 Preliminaries

Formally, we define the encoding process as follows. Let the full video stream be V^T with T total frames. For each video frame f_t at time t, let Q_t^i denote the query feature at layer i, and let $K_t^i \in \mathbb{R}^{N\times D}$ and $V_t^i \in \mathbb{R}^{N\times D}$ denote the corresponding key and value features, where N is the number of tokens per frame and D is the feature dimension.

Let ω denote the sliding-window size in frames. At layer i, the sliding window consists of the key–value features from the previous ω frames, $W_t^i = \{K_j^i, V_j^i\}_{j=t-\omega-1}^{t-1}$ containing a total of ωN tokens. We define the concatenated window keys and values as

$$K_{W_t}^i = \mathrm{concat}(\{K_j^i\}_{j=t-\omega-1}^{t-1}), \quad V_{W_t}^i = \mathrm{concat}(\{V_j^i\}_{j=t-\omega-1}^{t-1}).$$

The representation of frame f_t at layer i is then computed as

$$O_t^i = \mathrm{Attn}\big(Q_t^i,\ \big[K_{W_t}^i\,;\,K_t^i\big],\ \big[V_{W_t}^i\,;\,V_t^i\big]\big)\,.$$

For the next frame f_{t+1}, the key–value features K_t^i and V_t^i are appended to the sliding window W_{t+1}^i and key–value pair $\{K_{t-\omega-1}^i, V_{t-\omega-1}^i\}$ is removed and offloaded. At this stage, we average-pool

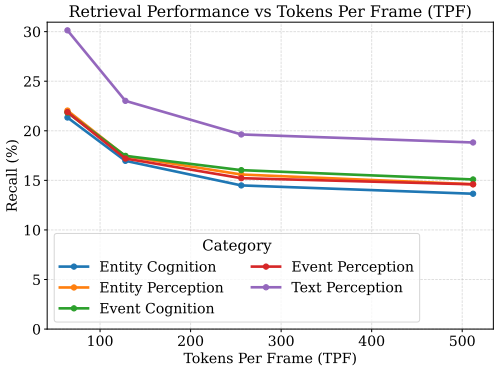


Figure 2: Increasing per-frame token budget leads to substantial declines in average layer-wise recall across a variety of different questions.

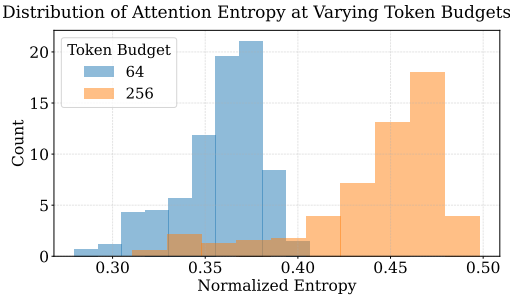


Figure 3: Normalized entropy for 677 sliding windows under 2 separate token budgets. At the higher token budget, sliding window attention tends to exhibit higher entropy, suggesting a struggle to focus on relevant frames.

each frame feature to form a representative vector $\mathbf{k}^i_{t-\omega-1} = \frac{1}{N}\sum_{n=1}^{N} K^i_{t-\omega-1}[n]$. These vectors are stored on GPU.

During question answering, let $\mathcal{Q}^i \in \mathbb{R}^{N_q \times D}$ denote the question embedding at layer i, where N_q is the number of tokens in the question. We compute a question representation $\mathbf{q}^i = \frac{1}{N_q}\sum_{n=1}^{N_q} \mathcal{Q}^i[n]$ for frame retrieval. Query–frame scores $S_{\text{internal}} \in \mathbb{R}^T$ are computed via cosine similarity between $\mathbf{q}^i$ and the set of representative frame vectors $\{\mathbf{k}^i_j\}_{j=1}^T$.

Let R denote the indices of the top-k retrieved frames. The corresponding key–value pairs are then retrieved for question answering. Using the retrieved keys K^i_R and values V^i_R, the output is computed as

$$O^i = \text{Attn}\big(Q^i,\ [K^i_R;\, K^i],\ [V^i_R;\, V^i]\big)\,.$$

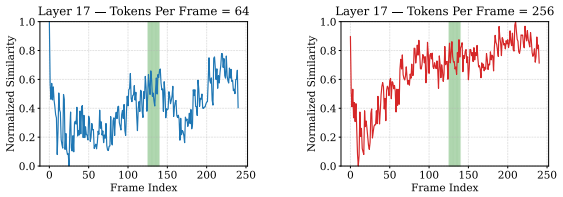
4 Analysis

In this section, we analyze how scaling per-frame token budgets affects model performance for dynamic resolution models. Notably, we find that naively scaling the per-frame token budget counterintuitively degrades both retrieval quality and downstream question-answering performance in KV-cache-based methods, despite the intuition that richer visual representations should improve understanding.

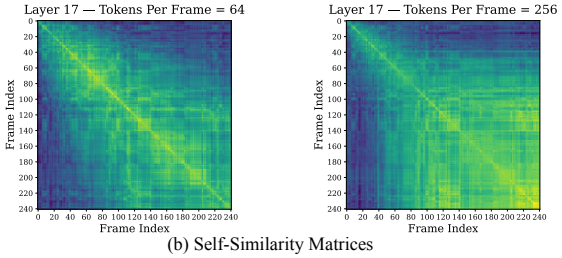
4.1 Scaling Token Budget Hurts Performance

Prior work on KV-cache compression has largely been applied to models with static token budgets, such as LLaVA-OneVision or LLaVA-Video. In contrast, few studies integrate KV-cache memory for dynamic-resolution models such as Qwen2.5-VL (41). We integrate Qwen2.5-VL with KV-cache encoding and retrieval, and apply it to CG-Bench, which provides the "clue" frames required to answer the question. This allows us to examine not only whether the model answers the question correctly, but also whether the model retrieves relevant frames from its KV-cache. All experiments use 128 input frames. Since Qwen2.5-VL represents two frames with one feature, this is equivalent to 64 frame-wise features.

Retrieval quality decreases as we increase the token budget. In Figure 2, we illustrate how increasing per-frame token budget impacts average layer-wise recall performance. We observe a declining trend, where increasing tokens per frame consistently leads to a drop in average recall. We confirm this trend across different question types, such as entity and text perception. Notably, increasing the number of tokens from 64 to 128 leads to a $\sim$5% drop in recall across all question categories. On average, increasing token budget from 64 to 512 tokens results in a 7% drop in recall.



(a) Query-Frame Scores



(b) Self-Similarity Matrices

Figure 4: In (a), we identify a systematic trend where query-frame similarity scores progressively increase across the video. In (b), self-similarity maps of the key representations become more redundant as we increase tokens per frame.

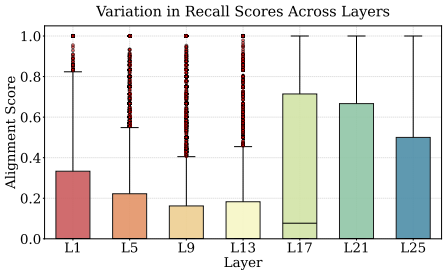


Figure 5: Recall for retrieving CG-Bench "clue" frames shows high variance across layers and is generally low.

Method	Num Tokens	CG-Bench	LVBench
Qwen2.5-VL-7B	64	34.27	**37.25**
+ ReKV (0.5 FPS → 128)	64	**35.23**	37.06
Qwen2.5-VL-7B	128	**37.02**	**39.19**
+ ReKV (0.5 FPS → 128)	128	37.00	36.60
Qwen2.5-VL-7B	256	**38.43**	**41.51**
+ ReKV (0.5 FPS → 128)	256	37.45	37.96

Table 1: Question-answering performance with KV-cache memory (ReKV) degrades at higher token budgets. (0.5 FPS → 128) denotes sampling 128 frames at 0.5 FPS.

Retrieval at higher token budgets fails due to temporal bias. We plot the similarity scores for frames encoded at 64 tokens per frame and at 256 tokens per frame, labeling the ground-truth segment in green (Figure 4, top). For frames encoded at lower token budget, the similarity score peaks at the ground-truth location, indicating successful retrieval. However, at 256 tokens per frame, the query-frame similarity progressively increases across the video, biasing frame selection toward the end of the video.

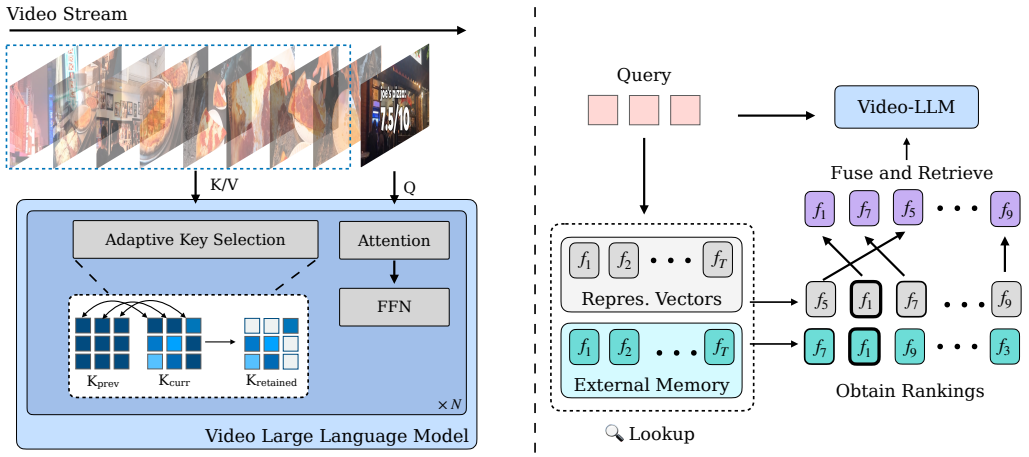
Self-similarity matrices confirm existence of temporal bias. Since query–frame retrieval depends critically on the quality of the key representations stored in the KV-cache, we analyze the self-similarity of representative frame vectors under different per-frame token budgets. We compute frame-wise self-similarity matrices for 64 and 256 tokens per frame, and visualize representative examples in Figure 4 (bottom). Strikingly, at higher token budgets, representative vectors from different frames become substantially more similar to one another, indicating increased redundancy and reduced discriminability.

Sliding window attention is less selective at higher token budgets. We hypothesize that at higher token budgets, the attention mechanism is unable to distinguish relevant spatiotemporal information from each frame. To validate this, we measure the normalized entropy of the attention scores in the sliding window during video encoding for both a token budget of 64 and 256. The distribution over 677 sliding windows (Figure 3) confirms that at higher token budget, sliding window attention exhibits greater entropy, indicating less selective behavior. This leads to degraded feature retrievals, which cascades into failures in question answering.

Increasing per-frame token budget deteriorates question-answering capabilities. As detailed in Table 1, at the lowest token budget of 64, uniform-retrieval performance on CG-Bench exceeds the uniform sampling setting by +0.95% and is roughly equivalent on LVBench. However, at 128 tokens per frame, we observe significant degradation on LVBench with a 2.59% drop. This trend persists at 256 tokens per-frame, with a 0.98% drop on CG-Bench and a 3.55% drop on LVBench.

4.2 Layer-wise Internal Retrieval is Unreliable

ReKV introduces both internal and external mechanisms for retrieving relevant video information from the KV-cache. Internal retrieval relies on the pre-trained MLLM's internal attention maps for



(a) Adaptive Key Selection

(b) Retrieval Ensemble

Figure 6: (a) Our model processes the actual video stream a single time only to encode video features. We perform adaptive key selection to construct a sparse KV-cache. (b) MemStream uses matching with the question tokens to retrieve the relevant K/V features from the cache. Since we select features on a frame-wise basis, we can perform this retrieval with arbitrary vision-language models. We leverage this to construct a training-free retrieval ensemble for the retrieval.

measuring the relevance of each stored frame to the question. Alternatively, external retrieval uses a pre-trained vision-language model such as CLIP to generate query-frame scores. ReKV contends that internal retrieval is more robust, with each layer acting as an independent expert. To isolate from the effect of larger token budgets, all analysis here uses 64 tokens per frame. We assess each layer's effectiveness using its recall score, shown for a selection of layers in Figure 5. We observe two notable trends: recall performance drastically changes depending on the layer, and it is highly variable even within a single layer. Some layers have a median recall of 0, indicating that the majority of the time they are unable to retrieve any relevant frames. This underscores the need for a more stable retrieval mechanism that can complement layer-wise information.

5 Approach

We introduce MemStream, a training-free unified framework for scalable processing and retrieval of dense video streams. As shown in Figure 6, our approach is split into two stages: encoding and retrieval. During encoding, we replace dense sliding window attention with Adaptive Key Selection (AKS), a sparse compression and selection strategy to identify and preserve critical video information in the sliding window. This has a twofold benefit: it preserves local spatiotemporal information across frames, improving KV-cache quality for both retrieval and question-answering, and it reduces latency and memory usage. When retrieving from the KV-cache memory, we propose a retrieval ensemble strategy that leverages external models to aid and improve internal retrieval quality.

5.1 Adaptive Key Selection for Sparse Sliding-Window Attention

Building on our analysis in Section 4.1, we propose an adaptive compression and selection strategy for sparse sliding-window attention. We emphasize that this selection occurs only for the attention process, while the full key features are stored and processed in the KV-cache. Recall that the sliding window at layer i is defined as $W_t^i = \{(K_j^i, V_j^i)\}_{j=t-\omega-1}^{t-1}$. Our goal is to compress and select a representative subset $W_{\text{comp}}^i \subset W_t^i$ that captures its critical spatiotemporal information. Adaptive Key Selection (AKS) identifies and eliminates temporal redundancy in the sliding window. Concretely, for each pair of adjacent key features K_t^i and K_{t-1}^i, we wish to only retain the patches that are most unique to K_t^i. To this end, we compute patch-wise cosine similarity between corresponding spatial tokens and select the top-k least similar (i.e., most distinctive) patch features from K_t^i, where k is

fixed. We note that this selection process occurs before applying positional embeddings, ensuring that the redundancy calculation is only done based on shared semantics of the patch features.

5.2 KV-Cache Retrieval via Retrieval Ensemble

While ReKV explores external retrievers such as CLIP for query–frame retrieval, it ultimately discards these in favor of purely internal retrieval. We contend that internal and external retrieval strategies are complementary and can mutually reinforce one another. Furthermore, external signals can help stabilize internal signals for more consistent layer-wise performance.

In particular, intermediate KV features encode rich contextual information accumulated across the video stream, but may lack fine-grained spatial or motion-specific cues. Conversely, frame- or clip-level features extracted from vision–language encoders often capture salient semantic details, yet lack access to broader temporal context.

To this end, we propose a training-free retrieval ensemble design that fuses internal attention-based signals with external vision model retrieval using reciprocal rank fusion (RRF). This approach allows strong retrieval signals from one expert to compensate for weaker signals from another, while incorporating complementary information from both internal and external representations.

In the internal retrieval strategy, layer-wise query-frame scores $S^i_{\text{internal}} \in \mathbb{R}^T$ are computed between the question embedding at layer i and the stored representative frame vectors $\{\mathbf{k^i_j}\}_{j=1}^T$. We follow a similar procedure when using an external vision-language model E. We denote E_{vis} as the visual encoder and E_{text} as the corresponding text encoder. While encoding, we compute a frame-wise feature $x_t \in \mathbb{R}^{1\times d} = E_{\text{vis}}(f_t)$ for each frame f_t in the video stream. During question-answering, we compute question Q's embedding $q = E_{\text{text}}(Q)$ and the cosine-similarity between q and all frame embeddings to obtain $S_{\text{external}} \in \mathbb{R}^T$.

Rather than fusing raw scores, we take inspiration from information retrieval literature and utilize reciprocal-rank fusion (RRF) (44). Formally, let

$$R^i = \{R^i_{\text{internal}},\ R_{\text{external}}\}$$

denote the set of retrieval rankings at layer i, where R^i_{internal} is the ranking produced by internal retrieval and R_{external} is the ranking produced by external retrieval. The RRF score for frame f_t at layer i is defined as

$$\text{RRFScore}^i(t) = \sum_{r\in R^i} \frac{1}{k + r(t)},$$

where $r(t)$ denotes the rank assigned to frame f_t by ranking r, and k is a fixed constant. This scoring function reinforces agreement between rankings while preventing outliers from having too large an effect. We select the top-k KV features from the cache at each layer using the computed RRF scores.

6 Experiments

6.1 Benchmark Datasets

We evaluate on a number of long-form video understanding benchmarks, including both offline and online benchmarks. Details are in Table 11 of the Appendix. For offline evaluation, we use the multiple-choice subset of CG-Bench (23), which evaluates a model's ability to retrieve relevant segments for question-answering, LVBench (47) for very long video understanding, and the "Long" subset of VideoMME (22), containing 300 videos with an average duration of 41 mins. For online evaluation, we use RVS-Ego and RVS-Movie (34), which evaluate streaming VQA via open-ended question-answering scored with the LLM-as-a-Judge paradigm.

6.2 Implementation Details

We integrate MemStream with LLaVA-OneVision-7B, InternVL3-8B, Qwen2.5-VL-3B/7B, and Qwen3-VL-8B. Following ReKV, videos are processed at 0.5 FPS with $\sim$256 tokens per frame. For Qwen2.5-VL, we set a minimum of 200 tokens under dynamic resolution, a 17K-token sliding window (roughly 64–68 frames), and 64 retrieved frame features for question answering. For Qwen3-VL-8B, we evaluate with a token budget of approximately 512 tokens per frame, with a minimum

Table 2: **Offline VQA Results.** Effectiveness of MemStream for long video question answering. We highlight the best performance in **bold**. For VideoMME, we use the "Long" subset only. † indicates reproduced results using the available codebase.

Settings						Benchmarks		
Model	N_{tokens}	N_{frames}	Encoding	Retrieval	Train Free?	CG-Bench	LVBench	VideoMME
Offline Video Question Answering								
InternVL3-8B								
+ Uniform Sample	8K	32	N/A	N/A	✓	38.60	–	–
Qwen2.5-VL-7B								
+ Uniform Sample	17K	128	N/A	N/A	✓	38.43	41.51	55.67
+ Uniform Sample	68K	512	N/A	N/A	✓	41.48	40.03	52.11
Online Video Question Answering								
LLaVA-OneVision								
+ rLiVS (45)	10K	(0.5 FPS → 51)	Full	Internal	✓	33.10	–	–
+ ReKV (14)	15K	(0.5 FPS → 64)	Full	Internal	✓	33.90	39.90	**47.22**
+ MemStream (Ours)	15K	(0.5 FPS → 64)	AKS	Ensemble	✓	**35.93**	**44.93**	47.00
Qwen2-VL-7B								
+ Flash V-Stream (36)	12K	(1 FPS →≤ 12K tokens)	N/A	N/A	✗	–	42.00	–
VideoXL-2								
+ TimeScope (24)	–	(1 FPS →≤ 800)	–	–	✗	38.47	–	–
Qwen2.5-VL-3B								
+ ReKV (14)	17K	(0.5 FPS → 128)	Full	Internal	✓	32.59	38.54	47.78
+ MemStream (Ours)	17K	(0.5 FPS → 128)	AKS	Ensemble	✓	**39.31**	**46.16**	**50.44**
InternVL3-8B								
+ ReKV (14)	17K	(0.5 FPS → 128)	Full	Internal	✓	41.71	43.31	54.00
+ MemStream (Ours)	17K	(0.5 FPS → 128)	AKS	Ensemble	✓	**44.75**	**49.19**	54.00
Qwen2.5-VL-7B								
+ TimeChat-Online (31)	–	(1 FPS → 54% tokens)	N/A	N/A	✗	–	–	52.40
+ StreamAgent (46)	32 TPF	(1 FPS)	Full	Internal	✓	–	–	50.60
+ FluxMem (40)	64 TPF	(1 FPS)	N/A	N/A	✓	38.14†	44.22†	54.00
+ Hermes (41)	6K	(1 FPS)	N/A	N/A	✓	39.64†	38.86†	**54.44**
+ ReKV (14)	17K	(0.5 FPS → 128)	Full	Internal	✓	36.17	39.64	51.78
+ MemStream (Ours)	17K	(0.5 FPS → 128)	AKS	Ensemble	✓	**44.19**	**48.10**	54.22
Qwen3-VL-8B								
+ ReKV (14)	39K	(0.5 FPS → 128)	Full	Internal	✓	52.84	51.45	**59.44**
+ MemStream (Ours)	39K	(0.5 FPS → 128)	AKS	Ensemble	✓	**54.78**	**55.65**	59.11

Table 3: **StreamingVQA Results.** Evaluation of MemStream on RVS-Ego and RVS-Movie, measuring answer accuracy and quality, and runtime and memory usage. Following ReKV, we use GPT-3.5-Turbo as our LLM judge.

Settings			RVS-Ego		RVS-Movie		Running Speed		Memory Usage	
Model	Encoding	Retrieval	Acc	Score	Acc	Score	Video Enc.	Latency	GPU	KV-Cache
Qwen2.5-VL-7B										
ReKV (14)	Full	Internal	64.2	4.00	**61.6**	**3.65**	15.10 FPS	1.7s	29 GB	11.1 GB/h
MemStream (Ours)	AKS	Internal	**67.8**	**4.01**	59.1	3.60	12.44 FPS	1.6s	29 GB	11.1 GB/h
MemStream (Ours)	AKS	Ensemble	67.4	**4.01**	59.7	3.60	10.44 FPS	1.6s	32 GB	11.1 GB/h

of 482 tokens per frame, setting the sliding-window limit to 39K tokens. For external retrieval, we evaluate CLIP and PECore (48) with ViT-L backbones and set the reciprocal rank fusion constant k to 60 following (44).

6.3 Results

We show offline results in Table 2 and component-wise effects in Table 5. On Qwen2.5-VL, our method significantly outperforms ReKV and recent baselines HERMES (41) and FluxMem (40). AKS alone improves CG-Bench by 5.5% and LVBench by 4.1%. Adding our retrieval ensemble further improves performance by 2.4% on CG-Bench and 4.3% on LVBench. External retrieval also provides substantial gains, but our combined strategy surpasses it by 2.3% on both benchmarks. On VideoMME, our method improves over ReKV by about 2.5%. External retrieval alone slightly degrades performance, likely because VideoMME emphasizes holistic understanding, while external retrieval mainly helps find key frames. We show an example of the improved retrieval using a sample from CG-Bench in Figure 7. We next examine online accuracy, quality, latency, and memory usage

Table 4: **Encoding Strategy Ablations.** Comparison of patch-wise and frame-wise encoding strategies across benchmarks.

Encoding Type	Strategy	Comp. Rate	CG-Bench	LVBench	VideoMME
	Full	∼1×	36.17	39.64	51.78
Patch-wise	A.1	∼4×	40.02	40.41	51.78
	A.2	∼16×	41.15	42.93	52.56
	B.1	∼12×	**42.18**	43.06	52.89
	AKS	∼16×	41.63	**43.77**	**54.56**
Frame-wise	A.3	∼8×	**41.63**	42.35	**53.78**
	B.2	∼8×	40.48	42.41	52.33
	B.3	∼8×	41.22	**43.25**	52.22

Table 5: **Breakdown of Performance Gains.** AKS and external retrieval each independently improve performance; combining both with our retrieval ensemble achieves the best results.

Encoding	Retrieval	CG-Bench	LVBench	VideoMME
Full	Internal	36.17	39.64	51.78
AKS	Internal	41.63	43.77	**54.56**
Full	External	41.77	45.84	50.89
AKS	External	43.21	47.19	52.33
AKS	Ensemble	**44.19**	**48.10**	54.22

Table 6: **Retrieval Strategy Ablation.** Comparison of internal, external, and ensemble retrieval. We use the AKS encoding strategy and RRF fusion, with equal weights for the internal and external rankings.

Retrieval Strategy	External Model	CG-Bench	LVBench	VideoMME
Internal Only	–	41.63	43.77	54.56
External Only	CLIP	42.39	47.45	53.67
External Only	PECore	43.21	47.19	52.33
Ensemble (Ours)	CLIP	43.84	47.39	**55.22**
Ensemble (Ours)	PECore	**44.19**	**48.10**	54.22

Table 7: **Fusion Strategy Ablation.** Comparison of early- and late-fusion methods for combining internal and external retrieval signals. We use PE-Core as the external encoder and the AKS encoding strategy.

Fusion Method	CG-Bench	LVBench	VideoMME
L2-Concat	43.57	48.03	52.89
RRF (Ours)	**44.19**	**48.10**	**54.22**

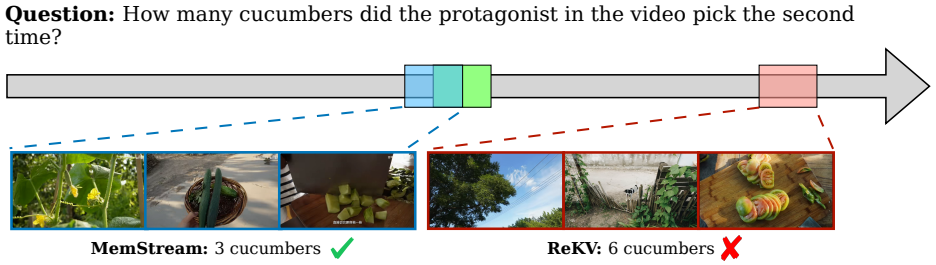


Figure 7: **Qualitative Example.** We compare ReKV's (red) and MemStream's (blue) respective retrievals and answers. The ground-truth segment is shown in green.

in Table 3. MemStream outperforms ReKV on RVS-Ego by 3.6% with minimal latency change, but drops by 2% on RVS-Movie, likely due to aggressive compression, as discussed in Section C.3.

6.4 Ablation Study

Impact of Encoding Strategy We explore sparse sliding-window attention designs methodically, evaluating static and dynamic selection and compression strategies at both patch- and frame-level granularity. For static strategies, we consider average pooling (**A.1**), which aggregates local spatial information within a $k \times k$ kernel, dilated sampling (**A.2**), which subsamples spatial tokens by selecting every k^{th} pixel feature along both height and width dimensions, and uniform frame sampling within the sliding window (**A.3**). For dynamic strategies, in addition to Adaptive Key Selection (**AKS**), we design three further compression variants: token-merging (**B.1**), inspired by ToME (49), which identifies and compresses redundant patches within each K_t^i without merging across time; frame-wise k-means clustering (**B.2**), which identifies representative centroid frames; and a temporal redundancy strategy (**B.3**), which retains only frames exhibiting the greatest change over time, computed as the average cosine similarity between adjacent key features. Table 4 shows results of these static and dynamic encoding strategies, validating our decision. AKS achieves the best overall results despite the highest compression rate. More ablation studies can be found in Sections C.4 and C.5.

Impact of Retrieval Strategy Table 6 shows that using CLIP or PECore for retrieval results in significant improvements for both CG-Bench and LVBench. Optimal results are achieved when combining PECore with internal retrieval via our retrieval ensemble. Table 7 shows that our reciprocal rank fusion (RRF) is superior to simple concatenation (L2-Concat) across all three offline benchmarks.

Table 8: **Impact of Scaling Token Budget.** AKS maintains performance over ReKV when scaling tokens per frame to higher budgets. *768 tokens per frame is evaluated on videos whose native resolution supports this budget without upsampling.

Tokens/Frame	Encoding	Comp. Rate	CG-Bench	LVBench	VideoMME
∼256	Full	1×	36.17	39.64	51.78
	AKS	∼16×	**41.63**	**43.77**	**54.56**
∼512	Full	1×	35.41	38.35	51.00
	AKS	∼16×	42.06	42.16	52.67
	AKS	∼25×	**42.13**	**43.32**	**53.22**
∼768*	Full	1×	35.55	35.91	50.06
	AKS	∼16×	41.37	39.55	**53.56**
	AKS	∼25×	**41.76**	**42.90**	52.56

Table 9: **Generalization Across Video-LLMs.** We observe that our gains generally transfer to LLaVA-OV and InternVL3.

		LLaVA-OV-7B		InternVL3-8B	
Encoding	Retrieval	CG-Bench	LVBench	CG-Bench	LVBench
Full	Internal	33.90	39.90	41.71	43.31
AKS	Internal	33.98	39.12	41.45	43.38
Full	External	35.49	43.51	**44.76**	47.84
AKS	External	35.62	44.29	44.66	49.00
AKS	Ensemble	**35.93**	**44.93**	44.75	**49.19**

Table 10: **Recall Across Video-LLMs.** AKS improves internal recall across all backbones.

Encoding	LLaVA-OV-7B	InternVL3-8B	Qwen2.5-VL-7B	Qwen3-VL-8B
Full	15.72	12.47	16.27	37.97
AKS	**18.14**	**13.47**	**28.91**	**38.67**

Impact of Token Budget Table 8 scales Qwen2.5-VL-7B from 256 to 768 tokens per frame. AKS outperforms full encoding at every budget on all three benchmarks, while full encoding degrades as the budget grows. Raising the compression rate to $\sim 25\times$ at 512 and 768 tokens helps mitigate the increased redundancy at these budgets and recovers most of the loss on CG-Bench and LVBench.

Impact across Video-LLMs Table 9 provides a breakdown of each component for LLaVA-OV-7B and InternVL3. Internal retrieval is the primary bottleneck on these backbones, so AKS sees limited gains. When retrieval is stabilized with an external encoder, AKS matches or surpasses full encoding. We show that AKS improves internal recall for all four backbone families in Table 10.

7 Limitations and Future Work

In this work, we identify two key insights: (1) sparse attention enables effective scaling to larger token budgets, and (2) external vision-language models can complement strong Video-LLMs for video understanding. These findings offer several directions for future work. First, our AKS strategy preserves a fixed ratio of patches; future work could make this ratio adaptive, retaining more patches for dynamic scenes and fewer for static ones. Second, training-based sparse methods may be more robust for scaling to larger token budgets. Finally, while we use fixed weights to combine internal and external retrievals, an adaptive mixture-of-experts-style routing could further improve performance.

8 Conclusion

We integrate KV-caching with modern MLLMs to enable efficient long video understanding. Through detailed analysis, we identify that higher token budgets in KV-cache streaming introduce temporal redundancy and retrieval bias. To address both, we propose Adaptive Key Selection (AKS) for higher-quality encoded features, and a training-free retrieval ensemble leveraging external model features for more precise temporal grounding. Our method, MemStream, achieves strong performance on both offline and online long video benchmarks.

9 Acknowledgements

This work was partially supported by NSF CAREER Award (#2238769). The authors acknowledge UMD's supercomputing resources made available for conducting this research. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or the U.S. Government. The authors would like to thank our colleagues Anubhav Gupta, Namitha Padmanabhan, and Max Ehrlich for their valuable conversations and feedback.

References

[1] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, pp. 34892–34916, 2023.

[2] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, *et al.*, "Llava-onevision: Easy visual task transfer," *arXiv preprint arXiv:2408.03326*, 2024.

[3] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[4] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, *et al.*, "Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution," *arXiv preprint arXiv:2409.12191*, 2024.

[5] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[6] J. Zhu, W. Wang, Z. Chen, Z. Liu, S. Ye, L. Gu, H. Tian, Y. Duan, W. Su, J. Shao, *et al.*, "Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models," *arXiv preprint arXiv:2504.10479*, 2025.

[7] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, *et al.*, "Gemma 3 technical report," *arXiv preprint arXiv:2503.19786*, 2025.

[8] B. He, H. Li, Y. K. Jang, M. Jia, X. Cao, A. Shah, A. Shrivastava, and S.-N. Lim, "Ma-lmm: Memory-augmented large multimodal model for long-term video understanding," 2024.

[9] Z. Cheng, S. Leng, H. Zhang, Y. Xin, X. Li, G. Chen, Y. Zhu, W. Zhang, Z. Luo, D. Zhao, and L. Bing, "Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms," 2024.

[10] S. Liu, C. Zhao, T. Xu, and B. Ghanem, "Bolt: Boost large vision-language model without training for long-form video understanding," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3318–3327, 2025.

[11] X. Tang, J. Qiu, L. Xie, Y. Tian, J. Jiao, and Q. Ye, "Adaptive keyframe sampling for long video understanding," *arXiv preprint arXiv:2502.21271*, 2025.

[12] L. Yao, H. Wu, K. Ouyang, Y. Zhang, C. Xiong, B. Chen, X. Sun, and J. Li, "Generative frame sampler for long video understanding," 2025.

[13] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, *et al.*, "Qwen2. 5-vl technical report," *arXiv preprint arXiv:2502.13923*, 2025.

[14] S. Di, Z. Yu, G. Zhang, H. Li, H. Cheng, B. Li, W. He, F. Shu, H. Jiang, *et al.*, "Streaming video question-answering with in-context video kv-cache retrieval," in *ICLR*, 2025.

[15] G. Sun, A. Singhal, B. Uzkent, M. Shah, C. Chen, and G. Kessler, "From frames to clips: Training-free adaptive key clip selection for long-form video understanding," 2025.

[16] M. Nie, D. Ding, C. Wang, Y. Guo, J. Han, H. Xu, and L. Zhang, "Slowfocus: Enhancing fine-grained temporal understanding in video llm," *Advances in Neural Information Processing Systems*, vol. 37, pp. 81808–81835, 2024.

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PmLR, 2021.

[18] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

[19] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, *et al.*, "Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features," *arXiv preprint arXiv:2502.14786*, 2025.

[20] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022.

[21] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[22] C. Fu, Y. Dai, Y. Luo, L. Li, S. Ren, R. Zhang, Z. Wang, C. Zhou, Y. Shen, M. Zhang, *et al.*, "Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24108–24118, 2025.

[23] G. Chen, Y. Liu, Y. Huang, Y. He, B. Pei, J. Xu, Y. Wang, T. Lu, and L. Wang, "Cg-bench: Clue-grounded question answering benchmark for long video understanding," *arXiv preprint arXiv:2412.12075*, 2024.

[24] X. Liu, M. Qin, Y. Shu, Z. Liang, Y. Tian, C. J. Zhang, B. Zhao, and Z. Liu, "Timescope: Towards task-oriented temporal grounding in long videos," *arXiv preprint arXiv:2509.26360*, 2025.

[25] H. Wang, Y. Nie, Y. Ye, Y. Wang, S. Li, H. Yu, J. Lu, and C. Huang, "Dynamic-vlm: Simple dynamic visual token compression for videollm," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20812–20823, 2025.

[26] K. Tao, C. Qin, H. You, Y. Sui, and H. Wang, "Dycoke: Dynamic compression of tokens for fast video large language models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18992–19001, 2025.

[27] Z. Liu, C.-W. Xie, P. Li, L. Zhao, L. Tang, Y. Zheng, C. Liu, and H. Xie, "Hybrid-level instruction injection for video token compression in multi-modal large language models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 8568–8578, 2025.

[28] S.-H. Lee, J. Wang, Z. Zhang, D. Fan, and X. Li, "Video token merging for long video understanding," *Advances in Neural Information Processing Systems*, vol. 37, pp. 13851–13871, 2024.

[29] X. Liu, Y. Shu, Z. Liu, A. Li, Y. Tian, and B. Zhao, "Video-xl-pro: Reconstructive token compression for extremely long video understanding," *arXiv preprint arXiv:2503.18478*, 2025.

[30] Z. Fan, K. Chen, R. Xing, Y. Li, L. Jiang, and Z. Tian, "Flashvid: Efficient video large language models via training-free tree-based spatiotemporal token merging," *arXiv preprint arXiv:2602.08024*, 2026.

[31] L. Yao, Y. Li, Y. Wei, L. Li, S. Ren, Y. Liu, K. Ouyang, L. Wang, S. Li, S. Li, *et al.*, "Timechat-online: 80% visual tokens are naturally redundant in streaming videos," in *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10807–10816, 2025.

[32] E. Song, W. Chai, S. Yang, E. Armand, X. Shan, H. Xu, J. Xie, and Z. Tu, "Videonsa: Native sparse attention scales video understanding," 2025.

[33] J. Chen, Z. Lv, S. Wu, K. Q. Lin, C. Song, D. Gao, J.-W. Liu, Z. Gao, D. Mao, and M. Z. Shou, "Videollm-online: Online video large language model for streaming video," in *CVPR*, 2024.

[34] H. Zhang, Y. Wang, Y. Tang, Y. Liu, J. Feng, J. Dai, and X. Jin, "Flash-vstream: Memory-based real-time understanding for long video streams," *arXiv preprint arXiv:2406.08085*, 2024.

[35] E. Song, W. Chai, G. Wang, Y. Zhang, H. Zhou, F. Wu, H. Chi, X. Guo, T. Ye, Y. Zhang, *et al.*, "Moviechat: From dense token to sparse memory for long video understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18221–18232, 2024.

[36] H. Zhang, Y. Wang, Y. Tang, Y. Liu, J. Feng, and X. Jin, "Flash-vstream: Efficient real-time understanding for long video streams," *arXiv preprint arXiv:2506.23825*, 2025.

[37] X. Zeng, K. Qiu, Q. Zhang, X. Li, J. Wang, J. Li, Z. Yan, K. Tian, M. Tian, X. Zhao, *et al.*, "Streamforest: Efficient online video understanding with persistent event memory," *arXiv preprint arXiv:2509.24871*, 2025.

[38] X. Chen, K. Tao, K. Shao, and H. Wang, "Streamingtom: Streaming token compression for efficient video understanding," *arXiv preprint arXiv:2510.18269*, 2025.

[39] M. Kim, K. Shim, J. Choi, and S. Chang, "Infinipot-v: Memory-constrained kv cache compression for streaming video understanding," *arXiv preprint arXiv:2506.15745*, 2025.

[40] Y. Xie, B. He, J. Wang, X. Zheng, Z. Ye, and Z. Wu, "Fluxmem: Adaptive hierarchical memory for streaming video understanding," in *CVPR*, 2026.

[41] H. Zhang, S. Yang, J. Fu, S.-K. Ng, and X. Qiu, "Hermes: Kv cache as hierarchical memory for efficient streaming video understanding," 2026.

[42] Y. Yang, Z. Zhao, S. N. Shukla, A. Singh, S. K. Mishra, L. Zhang, and M. Ren, "Streammem: Query-agnostic kv cache memory for streaming video understanding," 2025.

[43] Z. Ning, G. Liu, Q. Jin, W. Ding, M. Guo, and J. Zhao, "Livevlm: Efficient online video understanding via streaming-oriented kv cache and retrieval," 2025.

[44] G. V. Cormack, C. L. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 758–759, 2009.

[45] V. Dorovatas, S. Seifi, G. Gupta, and R. Aljundi, "Recurrent attention-based token selection for efficient streaming video-llms," *arXiv preprint arXiv:2510.17364*, 2025.

[46] H. Yang, F. Tang, L. Zhao, X. An, M. Hu, H. Li, X. Zhuang, Y. Lu, X. Zhang, A. Swikir, *et al.*, "Streamagent: Towards anticipatory agents for streaming video understanding," *arXiv preprint arXiv:2508.01875*, 2025.

[47] W. Wang, Z. He, W. Hong, Y. Cheng, X. Zhang, J. Qi, M. Ding, X. Gu, S. Huang, B. Xu, *et al.*, "Lvbench: An extreme long video understanding benchmark," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22958–22967, 2025.

[48] D. Bolya, P.-Y. Huang, P. Sun, J. H. Cho, A. Madotto, C. Wei, T. Ma, J. Zhi, J. Rajasegaran, H. Rasheed, *et al.*, "Perception encoder: The best visual embeddings are not at the output of the network," *arXiv preprint arXiv:2504.13181*, 2025.

[49] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," 2023.

[50] J. Niu, Y. Li, Z. Miao, C. Ge, Y. Zhou, Q. He, X. Dong, H. Duan, S. Ding, R. Qian, *et al.*, "Ovo-bench: How far is your video-llms from real-world online video understanding?," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18902–18913, 2025.

Going Down Memory Lane: Scaling Tokens for Video Stream Understanding with Dynamic KV-Cache Memory

Supplementary Material

Contents

A Further Preliminaries **15**

A.1 KV-Cache for Video Understanding 15

A.2 KV-Cache Size Calculation 15

B Hyperparameters and Settings **15**

B.1 Benchmark Datasets 15

B.2 Integrating KV-Cache Memory with Qwen2.5-VL 15

B.3 Compression Rates 16

C Experiments Cont. **16**

C.1 More Streaming Results 16

C.2 Inference Analysis of Retrieval Ensemble 16

C.3 RVS-Movie Discussion 17

C.4 Impact of Top-k and Sliding-Window Size 18

C.5 Impact of Compression Rate 18

C.6 Impact on Recall 18

C.7 Category Breakdown 18

C.8 VideoMME Performance 19

C.9 Results on Qwen3-VL-8B 20

C.10 Analysis of Layer-Wise Retrievals 21

C.11 More Visualizations 21

D Discussion **22**

D.1 Broader Impacts 22

A Further Preliminaries

A.1 KV-Cache for Video Understanding

Online Video Understanding. This work builds an adaptive memory framework for processing a continuous stream of video. This differs significantly from the conventional offline setting. Namely, frames are observed and stored in an incremental manner, preventing the model from using future context to encode current frames. Furthermore, the length of the stream is unknown, rendering uniform frame sampling strategies moot.

Vision-Language Understanding via KV-Cache. Modern large language models rely on key–value caching for improving efficiency during generation. There are two stages of this pipeline. First, the model performs prefilling; here, the model processes and stores intermediate key–value features of previous tokens, forming the KV-cache. Then, during decoding, each new token is generated by attending to the tokens in the stored KV-cache. MLLMs also exploit this structure for storing visual inputs in the KV-cache for use in downstream tasks such as image captioning or question-answering.

ReKV (14) extends this line of work by introducing a KV-cache-based memory for long-video processing. It incrementally encodes the video stream using sliding-window attention, offloading key–value features to RAM or disk once the frame has exited the window. During question-answering, ReKV leverages the internal attention of the LLM to retrieve relevant key–value entries from the cache.

A.2 KV-Cache Size Calculation

Assuming FP16 precision, the size of the KV-Cache is calculated as follows,

$$2 \times L \text{ layers} \times T \text{ frames} \times M \text{ tokens/frame} \times H \text{ heads} \times D \text{ dimension} \times 2 \text{ bytes}.$$

Qwen2.5-VL-7B has $L = 28$ layers with $H = 4$ heads of dimension $D = 128$. A 1-hour video processed at 0.5 FPS translates to $T = 900$ frames (using temporal-patch-size of 2). We set the frame-wise token budget between 200 and 256 tokens, resulting in a KV-cache size between 10.3 GB and 13.2 GB. Our results in Table 3 are within this bound.

B Hyperparameters and Settings

B.1 Benchmark Datasets

Table 11: **Dataset statistics.** We detail the number of videos, questions, and average length for each benchmark.

Dataset	# Videos	Avg. Length	# QA Pairs
CG-Bench (23)	1,219	27 min.	12,129
LVBench (47)	103	68 min.	1,549
VideoMME (Long) (22)	300	41 min.	900
RVS-Ego (34)	10	60 min.	1,465
RVS-Movie (34)	22	30 min.	1,905

B.2 Integrating KV-Cache Memory with Qwen2.5-VL

Integrating KV-Cache memory with Qwen2.5-VL requires three main considerations. First, Qwen2.5-VL applies dynamic resolution tokenization, where the number of tokens can vary for different images, and the original resolution of the image is preserved as best as possible. Second, rather than using a fixed 1D position encoding such as RoPE, Qwen2.5-VL employs the spatiotemporal-aware M-RoPE position encoding strategy. Lastly, Qwen2.5-VL uses a temporal patch-size of 2 during tokenization, such that one "frame-feature" corresponds to 2 frames. This means that for a video with N frames, the model stores $N_{kv} = N/2$ frame features.

B.3 Compression Rates

Throughout the paper, the compression rate r denotes the fraction of sliding-window keys that participate in attention during encoding. Patch-wise strategies, including AKS, retain N/r of the N patch features of each frame (i.e., $k = N/r$ in Section 5), while frame-wise strategies retain $1/r$ of the frames in the window. Compression only affects the keys and values attended to within the sliding window; the full KV features of every frame are still written to the cache and remain available for retrieval. For dynamic-resolution models, k is fixed while N varies across frames, so the rate is approximate (e.g., $\sim 16\times$ for Qwen2.5-VL at 200–256 tokens per frame). Table 12 lists the settings used for the main results. Unless stated otherwise, ablations in Section 6 and in this appendix use Qwen2.5-VL-7B at $\sim 16\times$. In the token-scaling study (Table 8) we additionally report $\sim 25\times$ at 512 and 768 tokens per frame, so that the number of retained keys per frame stays close to the 256-token setting. For Qwen3-VL-8B we use a mild $\sim 1.33\times$ rate (retaining 75% of patches), since higher rates reduce accuracy on that backbone (Section C.9).

Table 12: **Compression Settings for Main Results.** Token budget, sliding-window size, AKS compression rate, and the resulting number of retained keys per frame feature for each backbone in Table 2.

Backbone	Tokens/Frame	Sliding Window	AKS Rate	Retained Keys/Frame
Qwen2.5-VL-3B	~ 256	17K	$\sim 16\times$	~ 16
Qwen2.5-VL-7B	~ 256	17K	$\sim 16\times$	~ 16
InternVL3-8B	256	17K	$16\times$	16
LLaVA-OneVision-7B	196	15K	$4\times$	49
Qwen3-VL-8B	~ 512	39K	$\sim 1.33\times$	~ 384

C Experiments Cont.

All experimental results are conducted with Qwen2.5-VL-7B with 256 tokens per frame at 0.5 FPS, unless otherwise specified.

C.1 More Streaming Results

As our method focuses more on improving retrieval rather than real-time streaming understanding, we evaluate MemStream on the backward-tracing subset of OVOBench (50), which tests a streaming model's ability to recall past events via episodic memory (EPM), action sequence identification (ASI), and hallucination detection (HLD).

Table 13: **OVOBench Backward-Tracing Results.** We compare MemStream with ReKV along with other strong baselines. † indicates training-based methods

Model	Config	ASI	EPM	HLD	Avg
VideoLLM-online-8B† (37) (33)	–	22.2	18.8	12.2	17.7
Flash-VStream-7B† (37) (34)	–	39.1	37.2	5.9	27.4
Qwen2.5-VL-7B	ReKV	64.9	52.3	23.2	46.4
	MemStream	62.2	53.9	25.3	46.9

MemStream outperforms ReKV and other streaming baselines on average, with particular gains on episodic memory and hallucination detection. Together with our results on RVS-Ego (+3%) and offline benchmarks, these results demonstrate that MemStream's adaptive encoding and retrieval ensemble strategies consistently improve streaming video understanding across diverse settings.

C.2 Inference Analysis of Retrieval Ensemble

We first provide a breakdown of MemStream's runtime, explicitly identifying the cost of the external encoder. Table 14 shows that the external encoder accounts for 9.0% of total runtime (15.9% of encoding time) and adds about 6 GB of GPU memory, while external retrieval itself is negligible at 0.4%. Decoding and the MLLM visual backbone dominate the runtime.

Table 14: **Latency Breakdown with the External Encoder.** Wall-clock time per stage for Qwen2.5-VL-7B + MemStream (AKS encoding and PECore ensemble) on one 60-minute RVS-Ego video with 100 questions at 1 FPS on a single H200 GPU.

#	Stage	Time (s)	% of Runtime
1	MLLM visual backbone	101.6	27.5
2	External visual encoder (PECore)	33.2	9.0
2b	External retrieval	1.3	0.4
3	KV-cache encoding	73.9	20.0
4	Retrieval	14.2	3.8
5	Decoding	145.3	39.3
	Total	369.6	100

In the main paper, we showed that our plug-and-play design can be used to augment smaller VideoLLMs with strong external VL encoders. We investigate this further here.

Table 15: **MemStream on Smaller Models.** Qwen2.5-VL-3B + MemStream outperforms the larger Qwen2.5-VL-7B + ReKV on CG-Bench and LVBench.

Model	Config	CG-Bench	LVBench	VideoMME
Qwen2.5-VL-3B	ReKV	32.59	38.54	47.78
	MemStream	**39.31**	**46.16**	**50.44**
Qwen2.5-VL-7B	ReKV	36.17	39.64	51.78
	MemStream	**44.19**	**48.10**	**54.22**

Table 16: **Efficiency Comparison Across Model Sizes.** All configurations use AKS encoding.

Model	Retrieval	Video Enc.	Latency	GPU	KV-Cache
Qwen2.5-VL-7B	Internal	11.63 FPS	1.5s	29 GB	11.1 GB/h
Qwen2.5-VL-3B	Ensemble	10.58 FPS	0.7s	21 GB	7.1 GB/h
Qwen2.5-VL-7B	Ensemble	9.82 FPS	1.5s	32 GB	11.1 GB/h

Qwen2.5-VL-3B + MemStream outperforms the larger Qwen2.5-VL-7B + ReKV baseline on CG-Bench (39.31 vs 36.17) and LVBench (46.16 vs 39.64). This suggests that a smaller model equipped with MemStream can be a viable alternative to a larger model on resource-constrained devices.

C.3 RVS-Movie Discussion

In the main paper, we observed that RVS-Movie suffered a drop in performance at a $16\times$ compression ratio. We hypothesize that this may be due to AKS mistakenly discarding important semantics at high compression rates. To this end, we investigate model performance at a lower compression ratio on RVS-Movie (as shown in Table 17 and find that reducing the compression ratio from 1/16 to 1/4 boosts accuracy by 2%. Furthermore, we examine how AKS performs on content with subtle changes via a 1,067 video subset of CG-Bench containing TV and Music content, which is rich in dialogue and facial expressions. Examining Table 18, we find that AKS achieves strong gains over the ReKV baseline, suggesting it does not systematically discard important semantic content in low-dynamic scenes.

We disentangle whether AKS degrades the retrievals or the KV-cache representations used for QA by evaluating all four combinations of cache (which representations answer) x retrievals (which rankings select frames). Table 19 evaluates all four combinations of cache and retrieval. AKS retrievals improve accuracy for both cache representations, while the AKS cache is consistently lower than the ReKV cache, which localizes the degradation to the encoded representations. Failures concentrate on temporal-order questions; we hypothesize that discarding adjacent redundant patches at $16\times$ removes the subtle frame-to-frame changes that establish event order. For example, when asked what happens before two ships exchange fire, the ReKV cache correctly describes the crew's preparations, whereas the AKS cache misorders the events and drifts into generic scene description.

Table 17: **Effect of AKS Compression Rate on RVS-Movie.**

Comp. Rate	Acc	Score
$\sim 4\times$	61.1	3.64
$\sim 16\times$	59.1	3.60

Table 18: **AKS on Low-Dynamic Content (TV & Music Subset of CG-Bench).**

Encoding	TV & Music
Full	32.73
AKS	38.77

Table 19: **Cache vs. Retrieval on RVS-Movie.** Accuracy for all four combinations of which KV-cache answers the question (rows) and which retrieval ranking selects the frames (columns), using Qwen2.5-VL-7B at $\sim 16\times$ compression.

	ReKV Retrievals	AKS Retrievals
ReKV Cache	61.6	**63.4**
AKS Cache	58.0	59.1

C.4 Impact of Top-k and Sliding-Window Size

In Table 20, we provide ablations on the impact of the number of retrieved frames on model performance and compare ReKV (Full) and our strategy (AKS). We make two key observations. First, AKS achieves consistent performance gains across all top-k settings. Second, increasing the number of frames for retrieval, generally improves performance. We also ablate the impact of sliding window size on model performance in Table 21, observing that increasing sliding-window size generally boosts performance until a certain point, at which accuracy drops slightly. This could suggest increased redundancy, requiring greater compression.

Table 20: **Effect of Retrieval Size.**

Encoding	Top-k	CG-Bench	LVBench
Full	32	33.06	32.80
	64	34.54	34.80
	128	36.17	39.64
AKS	32	35.05	38.02
	64	38.43	39.64
	128	41.63	43.77

Table 21: **Effect of Sliding-Window Size.**

	8500	13000	17000	25000
LVBench	40.34	40.80	43.77	43.25

C.5 Impact of Compression Rate

In Table 22, we evaluate static selection patterns at compression rates $r \in \{4, 8, 16\}$. Patch-wise strategies retain $1/r$ tokens per frame, whereas frame-wise strategies retain $1/r$ frames from the sliding window. Sliding-window compression improves performance over full attention (up to +5.4% on CG-Bench and +3.3% on LVBench), with gains peaking at an intermediate compression factor. Additionally, in Table 23, we evaluate AKS at different compression ratios, observing that increasing the compression ratio generally improves performance.

C.6 Impact on Recall

In Table 24, we show how AKS and our external retrieval impact recall for different types of questions on CG-Bench.

C.7 Category Breakdown

In Tables 25 and 26, we present the breakdown of performance for various question types.

Table 22: **Effect of Sliding-Window Compression Rate.** We evaluate performance of static attention patterns at varying sliding-window compression rates.

Encoding	Comp. Rate	CG-Bench	LVBench	VideoMME
Full	1×	36.17	39.64	51.78
Pooled (A.1)	∼4×	40.02	**40.41**	**51.78**
	∼8×	39.81	40.35	50.33
	∼16×	39.54	39.57	50.22
Dilated (A.2)	∼4×	40.17	41.12	51.89
	∼8×	**41.69**	42.67	51.78
	∼16×	41.15	**42.93**	**52.56**
Uniform Sample (A.3)	∼4×	40.88	40.87	51.78
	∼8×	41.63	42.35	**53.78**
	∼16×	**41.83**	**43.19**	52.89

Table 23: **Effect of AKS Compression Rate.** We evaluate performance of AKS at varying compression ratios.

Comp. Rate	CG-Bench	LVBench
∼4×	40.66	43.12
∼9×	40.47	43.38
∼16×	41.63	43.77

Table 24: **CG-Bench Recall.** We show how frequently the top 64 retrievals overlap the features from the ground truth clue frames in CG-Bench.

Settings				CG-Bench Average Recall@64 Scores					
Model	Method	Encoding	Retrieval	Entity Cognition	Entity Perception	Event Cognition	Event Perception	Text Perception	Mean
Internal Retrieval									
Qwen2.5-VL-7B	Uniform	Uniform	Internal	20.57	22.20	21.98	22.29	26.12	22.24
	ReKV	Full	Internal	14.49	15.59	16.03	15.22	19.63	16.27
	MemStream	A.2 (×4)	Internal	19.97	21.42	20.86	20.23	27.31	21.59
	MemStream	A.2 (×16)	Internal	24.89	27.17	26.08	26.94	35.31	27.41
	MemStream	A.3 (×4)	Internal	21.83	23.64	22.38	21.84	29.95	23.36
	MemStream	A.3 (×16)	Internal	**26.85**	**29.51**	27.24	27.57	**38.32**	28.79
	MemStream	AKS (×4)	Internal	20.86	22.25	21.64	21.26	29.10	22.61
	MemStream	AKS (×16)	Internal	26.50	28.73	**27.35**	**28.44**	37.20	**28.91**
With External Retrieval									
PECore ViT-L	–	–	–	**54.54**	**63.96**	**52.43**	**54.51**	58.41	**55.94**
Qwen2.5-VL-7B	MemStream	AKS (×16)	Ensemble	51.00	58.04	48.84	51.02	**58.52**	52.32

C.8 VideoMME Performance

In the main paper, we observe that VideoMME performance sometimes experiences minimal improvement and even a slight drop. To understand this further, we investigate performance on the LLaVA-OV and InternVL3 backbones. Table 27 breaks down VideoMME by task type for InternVL3-8B, where MemStream matches ReKV in aggregate (Table 2). MemStream improves on tasks that require localizing specific events (action and object recognition and reasoning), but drops on tasks that require integrating information across the whole video (information synopsis and temporal reasoning). We attribute this to AKS retrievals concentrating on query-specific moments, which improves localization at the cost of broad coverage. LLaVA-OV-7B, where MemStream is likewise flat in aggregate, shows a similar pattern: gains on action and object recognition and losses or matching performance on temporal reasoning and information synopsis.

Table 25: **CG-Bench Breakdown.** We breakdown CG-Bench performance for a subset of question-types. We **bold** the best result and underline the second-best.

Settings				CG-Bench Accuracy					
Model	Method	Encoding	Retrieval	Entity Cognition	Entity Perception	Event Cognition	Event Perception	Text Perception	Mean
Internal Retrieval									
Qwen2.5-VL-7B	Uniform	Uniform	Internal	34.03	37.06	36.76	35.35	49.52	38.43
	ReKV	Full	Internal	33.12	36.35	34.73	31.36	45.72	36.17
	MemStream	A.2 (×4)	Internal	35.70	39.08	39.26	36.02	52.42	40.17
	MemStream	A.2 (×16)	Internal	36.61	40.14	39.26	**37.99**	54.36	41.15
	MemStream	A.3 (×4)	Internal	36.05	40.29	39.12	36.44	54.52	40.88
	MemStream	A.3 (×16)	Internal	**38.56**	40.14	**40.07**	37.32	**56.87**	**41.83**
	MemStream	AKS (×4)	Internal	36.96	39.98	39.66	36.15	52.50	40.66
	MemStream	AKS (×16)	Internal	38.08	**40.66**	39.86	37.70	55.01	41.63
With External Retrieval									
Qwen2.5-VL-7B	MemStream	AKS (×16)	PECore Only	38.77	43.25	40.74	38.12	59.45	43.21
	MemStream	AKS (×16)	Ensemble	**40.03**	**43.32**	**43.78**	**40.01**	**60.18**	**44.19**

Table 26: **LVBench Breakdown.** We breakdown LVBench performance across all question-types. We **bold** the best result and underline the second-best.

Settings				LVBench Accuracy						
Model	Method	Encoding	Retrieval	Entity Recog.	Event Understand.	Key Info. Retrieval	Reasoning	Summarizing	Temporal Ground.	Mean
Internal Retrieval										
Qwen2.5-VL-7B	Uniform	Uniform	Internal	41.65	39.72	42.96	42.29	**43.10**	39.09	41.51
	ReKV	Full	Internal	38.55	39.57	43.30	40.30	29.31	35.00	39.64
	MemStream	A.2 (×4)	Internal	40.33	41.11	44.33	41.29	31.03	36.82	41.12
	MemStream	A.2 (×16)	Internal	41.80	**42.35**	45.02	41.79	34.48	39.09	42.93
	MemStream	A.3 (×4)	Internal	40.47	39.88	45.36	43.28	27.59	36.82	40.87
	MemStream	A.3 (×16)	Internal	**43.13**	40.19	48.11	40.30	34.48	**40.45**	43.19
	MemStream	AKS (×4)	Internal	42.10	40.65	48.45	**46.27**	37.93	40.00	43.12
	MemStream	AKS (×16)	Internal	42.98	41.27	**50.52**	42.79	34.48	**40.45**	**43.77**
With External Retrieval										
Qwen2.5-VL-7B	MemStream	AKS (×16)	PECore Only	**51.26**	40.49	**58.42**	41.79	**36.21**	40.91	47.19
	MemStream	AKS (×16)	Ensemble	50.22	**43.89**	56.70	**44.28**	**36.21**	**45.00**	**48.10**

Table 27: **VideoMME Breakdown.** Per-task accuracy on the VideoMME Long subset for LLaVA-OV-7B and InternVL3-8B with ReKV and MemStream. The number of questions per task is in parentheses.

	LLaVA-OV-7B		InternVL3-8B	
Task Type	ReKV	MemStream	ReKV	MemStream
Object Reasoning (240)	44.6	**45.0**	55.0	**56.2**
Action Reasoning (180)	**47.2**	45.0	55.6	**56.7**
Information Synopsis (163)	67.5	67.5	**73.0**	66.9
Temporal Reasoning (91)	**37.4**	31.9	**35.2**	30.8
Action Recognition (63)	30.2	**38.1**	41.3	**50.8**
Object Recognition (54)	44.4	**48.1**	53.7	**57.4**

C.9 Results on Qwen3-VL-8B

In this section, we investigate how AKS interacts with the newer Qwen3-VL-8B backbone. Table 28 shows that scaling the token budget still degrades recall, from 39.46 to 36.80, although more weakly than on Qwen2.5-VL (Section 4) and without a corresponding drop in accuracy.

Table 29 varies the AKS compression rate at $\sim$512 tokens per frame. Unlike Qwen2.5-VL, where compressing the sliding window by $16\times$ improves accuracy (Table 8), Qwen3-VL-8B is on par with full encoding at $\sim 1.33\times$ and drops below it at $\sim 4\times$, even though recall continues to improve. This mirrors our RVS-Movie analysis (Table 19): AKS retrievals remain beneficial, but over-compressed cache representations hurt answering. It is also consistent with the much higher internal recall of Qwen3-VL-8B (37.97 vs. 16.27 for Qwen2.5-VL-7B, Table 10): the cache is far less redundant, so there is little for AKS to remove. We therefore use the $\sim 1.33\times$ setting for the Qwen3-VL-8B rows in Table 2.

To understand the drop in performance at $\sim 4\times$ compression, we restrict AKS to the first 12 layers and observe that this retains almost all of the recall gain (39.51 vs. 39.68) while matching full-encoding accuracy (52.99 vs. 52.84), indicating that the accuracy loss comes from compressing later layers, whose representations are less redundant, while early layers can be compressed safely. This is consistent with retrieval signal concentrating in later layers (Figure 8). This motivates future work on adaptive compression rates.

Table 28: **Token Scaling on Qwen3-VL-8B.** CG-Bench Recall@64 with full encoding (no compression) as the per-frame token budget grows.

Tokens/Frame	CG-Bench Recall@64
144–160	**39.46**
482–536	37.97
700–768	36.80

Table 29: **AKS Compression Rate on Qwen3-VL-8B.** With a less redundant cache, mild compression matches full encoding and higher rates degrade accuracy. All rows use internal retrieval at $\sim$512 tokens per frame.

Encoding	Comp. Rate	CG-Bench Recall	CG-Bench	LVBench
Full	1×	37.97	52.84	51.45
AKS	∼1.33×	38.67	**53.10**	**51.58**
AKS	∼4×	**39.68**	52.39	50.55

Table 30: **Layer-Banded AKS on Qwen3-VL-8B.** Restricting $\sim$4× AKS to the first 12 layers keeps almost all of the recall gain without the accuracy loss of all-layer compression. All rows use internal retrieval at $\sim$512 tokens per frame.

Encoding	Comp. Rate	Layers	CG-Bench Recall	CG-Bench	LVBench
Full	1×	–	37.97	52.84	51.45
AKS	∼1.33×	All	38.67	**53.10**	51.58
AKS	∼4×	All	**39.68**	52.39	50.55
AKS	∼4×	0–11	39.51	52.99	**51.65**

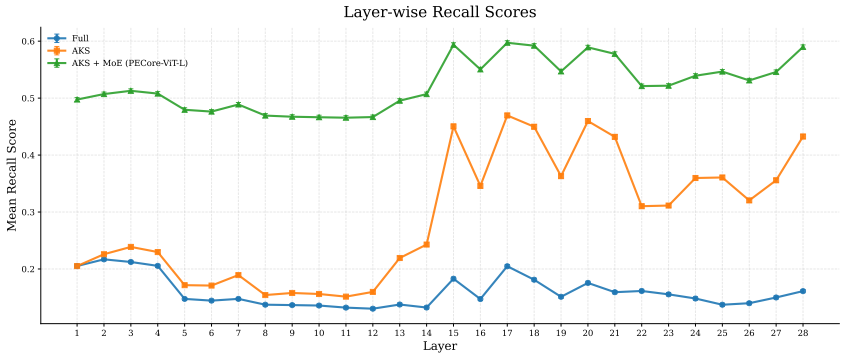


Figure 8: We plot layer-wise average recall scores for full sliding-window attention, AKS, and AKS combined with retrieval ensemble with PECore as the external retriever.

C.10 Analysis of Layer-Wise Retrievals

We plot average layer-wise recall scores for full, AKS, and AKS + Ensemble using the CG-Bench dataset in Figure 8. AKS significantly improves layer-wise recall for later layers, which is further improved by our retrieval ensemble strategy.

C.11 More Visualizations

We show layer-wise query-frame scores at 64 tokens per frame in Fig 9 and 256 tokens per frame in Fig 10. The ground-truth segment is highlighted in red.

We show layer-wise self-similarity matrices at 64 tokens per frame in Fig 11 and 256 tokens per frame in Fig 12. We observe that at 256 tokens per frame, there is higher redundancy in the self-similarity matrices across all layers.

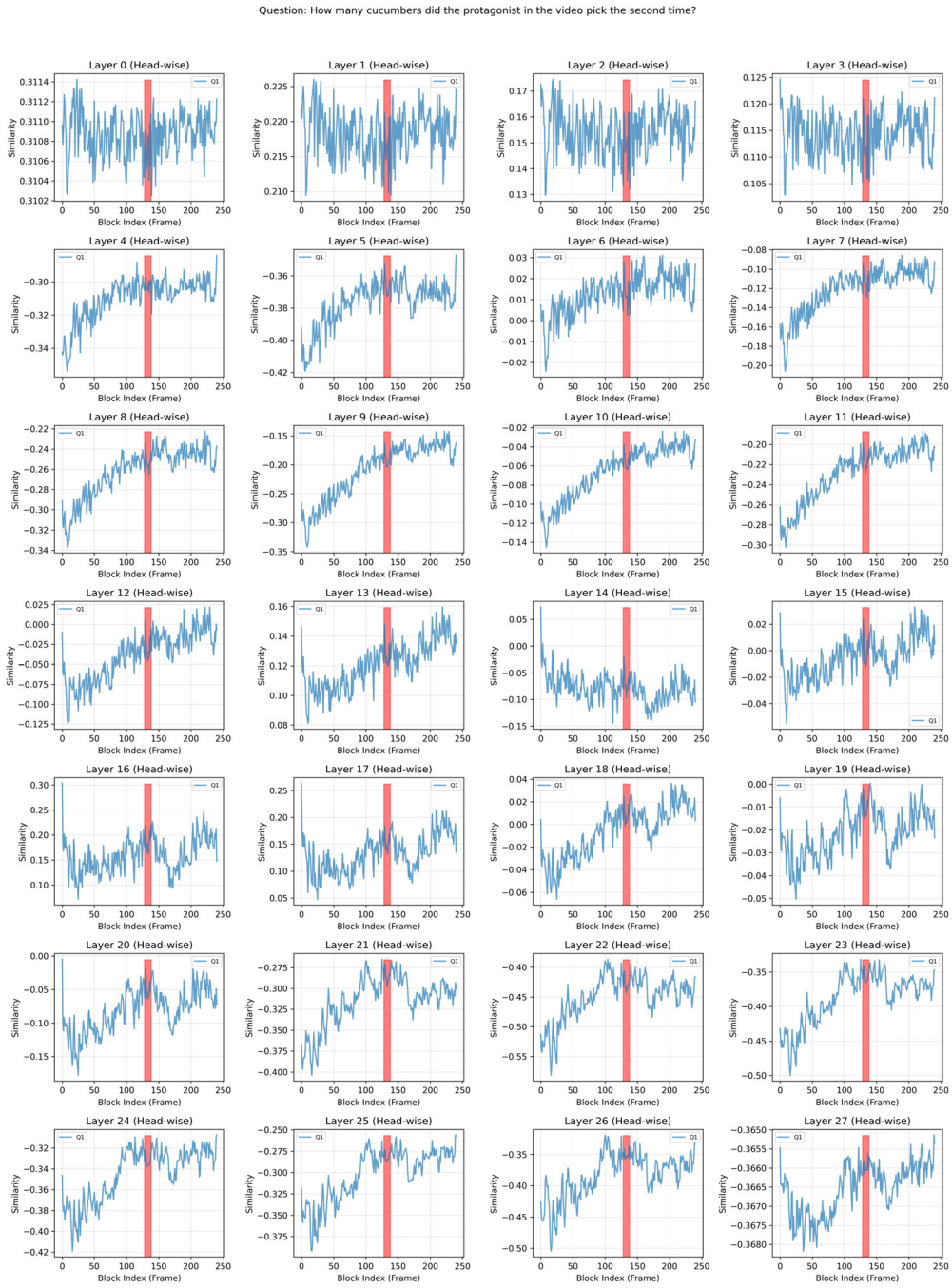


Figure 9: Layer-Wise Query-Frame Scores 64 Tokens Per Frame @ 241 Frames

D Discussion

D.1 Broader Impacts

Generally, the use of Video-LLMs has potential risks with respect to hallucinating content, and as such, we emphasize caution when employing such models in real-world scenarios. The streaming setting has seen recent applications for smart glasses and surveillance. Although our method does not directly store images of persons or objects, we acknowledge potential privacy concerns.

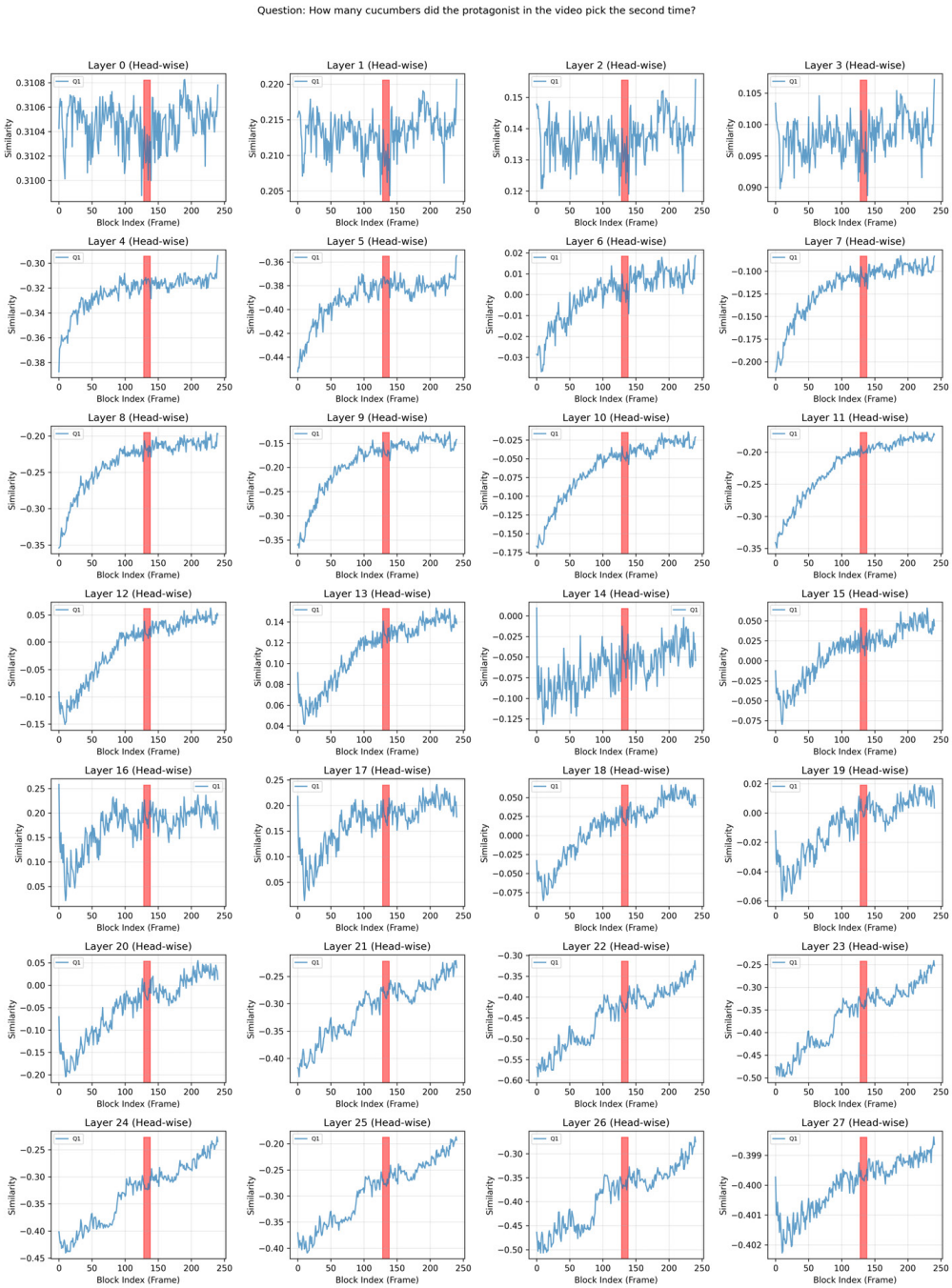


Figure 10: Layer-Wise Query-Frame Scores 256 Tokens Per Frame @ 241 Frames

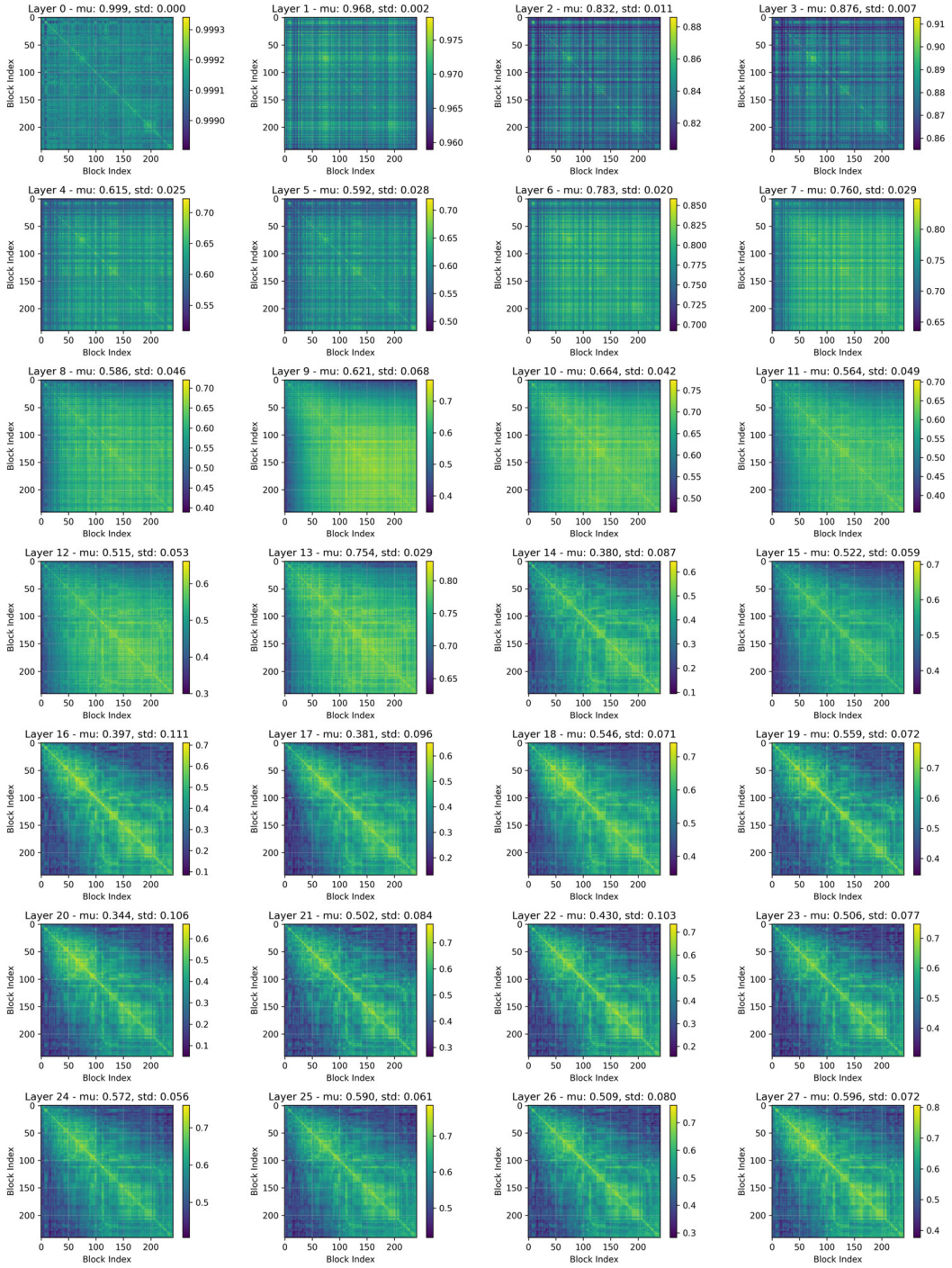


Figure 11: Layer-Wise Self-Similarity Matrices at 64 Tokens Per Frame @ 241 Frames

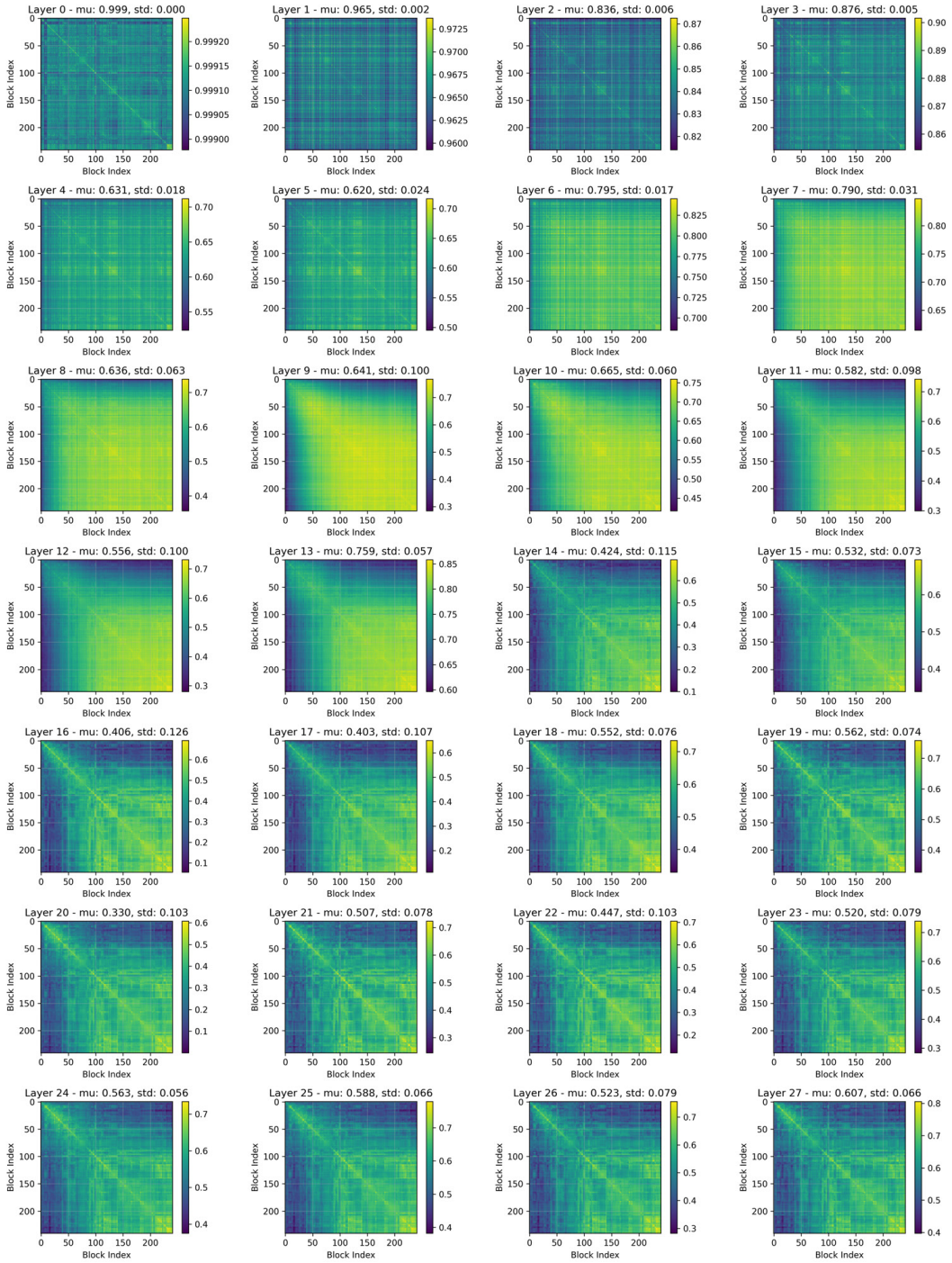


Figure 12: Layer-Wise Self-Similarity Matrices at 256 Tokens Per Frame @ 241 Frames

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and introduction clearly reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The supplementary discusses the limitations of the work. (Section 7)

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: There are no theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper does provide all information to reproduce the main experimental results. Additional details about the changes needed for adopting KV-cache encoding and retrieval can be found in the supplementary. (Section 5 and Section B.2)

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: At this time, we are unable to provide code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Yes all test details are provided, see Section 6.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Due to computational costs, we were unable to report error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Due to internal requirements, no details about the compute resources can be provided at this time.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and declare no deviation from it. We address concerns with Video-LLMs here, namely that these models are still in development and should be used cautiously.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, these are discussed in Section D.1.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We are not releasing any data or models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all owners of assets used in this paper, including models and datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not introduce any new assets in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing nor research with human subjects was conducted.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core development in this research does not involve any LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.