InfiniPot-V: Memory-Constrained KV Cache Compression for Streaming Video Understanding

Minsoo Kim[1†] **Kyuhong Shim**[2] **Jungwook Choi**[1‡] **Simyung Chang**[3‡]
[1]Hanyang University [2]Sungkyunkwan University
[3]Qualcomm AI Research, Qualcomm Korea YH[§]
{minsoo2333, choij}@hanyang.ac.kr khshim@skku.edu
simychan@qti.qualcomm.com

Abstract

Modern multimodal large language models (MLLMs) can reason over hour-long video, yet their key–value (KV) cache grows linearly with time—quickly exceeding the fixed memory of phones, AR glasses, and edge robots. Prior compression schemes either assume the whole video and user query are available offline or must first build the full cache, so memory still scales with stream length. InfiniPot-V is the first training-free, query-agnostic framework that enforces a hard, length-independent memory cap for *streaming* video understanding. During video encoding it monitors the cache and, once a user-set threshold is reached, runs a lightweight compression pass that (i) removes temporally redundant tokens via Temporal-axis Redundancy (TaR) metric and (ii) keeps semantically significant tokens via Value-Norm (VaN) ranking. Across four open-source MLLMs and four long-video and streaming-video benchmarks, InfiniPot-V cuts peak GPU memory by up to 94%, sustains real-time generation, and matches or surpasses full-cache accuracy—even in multi-turn dialogues. By dissolving the KV cache bottleneck without retraining or query knowledge, InfiniPot-V closes the gap for on-device streaming video assistants.

1 Introduction

Recent advances in multimodal large language models (MLLMs) have dramatically expanded the scope of visual reasoning. Vision–language instruction tuning now allows a single backbone to answer open-ended questions over long video sequences [27, 30, 48], while context-extension techniques such as FlashAttention-2 and RingAttention push the effective window into the million-token regime [7, 26, 33]. These breakthroughs underpin a new generation of *streaming video assistants* and *humanoid robots* that promise continuous, real-time scene understanding on mobile phones, AR glasses and edge robots [14, 31, 42, 37].

Streaming video understanding (SVU) diverges from conventional offline video understanding (OVU). Offline models see the entire clip and user query before inference, so they can tailor every compression or retrieval step. In streaming, frames arrive incrementally and future queries are unknown, forcing all pre-query processing to be *query-agnostic*. In addition, device memory is fixed, yet the transformer emits hundreds of tokens per frame, so the key–value (KV) cache grows linearly. For example, a 15-min, 10 fps clip processed by LLaVA-Next-Video-7B already needs demands ~ 100 GB of KV storage, far beyond the tens of gigabytes available on mobile or robotic platforms [52, 19].

[†]Work done during an internship at Qualcomm Technologies, Inc.
[‡]Corresponding authors.
[§]Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Prior work tackles long-video memory constraints at three stages (Fig.1). *Frame Sampling* [10] drops frames before encoding, reducing memory but severely degrading temporal coverage and accuracy. *Input-Vision Compression (IVC)* [36, 40] prunes redundant vision tokens after encoding, lowering Prefill load but still requiring the full vision token set to be stored in memory. *KV cache Compression (KVC)* [24, 12] selects query-relevant tokens after the Prefill step, offering the highest accuracy but only after materializing the full KV cache. The challenge intensifies in streaming scenarios: memory usage for Frame Sampling, IVC, and KVC grows almost linearly with video length, eventually exceeding device limits. KV cache offloading (e.g., ReKV [35]) expands memory space yet incurs costly data transfer, repeated for each query. Thus, no existing approach delivers the key property SVU needs: a *length-independent* and *query-agnostic* streaming video compression.

A natural approach to address memory constraints in streaming video is to exploit the strong *spatiotemporal redundancy* of video streams. We introduce **InfiniPot-V**, the first framework specifically designed for memory-constrained SVU. InfiniPot-V is *training-free*, *query-agnostic*, and operates *continuously* during inference. When the KV cache reaches a user-defined memory threshold M, it performs an in-place compression that frees space for new frames while preserving the semantic essence of prior context. This compression is guided by two lightweight and complementary metrics. *Temporal-axis Redundancy (TaR)* models Key embeddings as a 3D tensor over time and removes tokens with high cosine similarity to recent frames, effectively pruning static or repetitive content. *Value-Norm Importance (VaN)* ranks the remaining tokens by the ℓ_2 norm of their Value vectors—a strong, model-agnostic proxy for semantic salience—and applies a layer-adaptive pooling strategy. This compression is highly efficient, adding negligible latency while strictly enforcing memory limits.

Extensive evaluation confirms the effectiveness of this design. Across four open-source MLLMs (3B and 7B) and six long-video benchmarks—covering both offline (VideoMME, EgoSchema, MLVU, LongVideoBench) and streaming (RVS-Ego/Movie, OVO-Bench, StreamingBench) tasks—InfiniPot-V reduces input context length usage to as low as 6K for 50K-token contexts, with accuracy matching or exceeding full-cache baselines. It maintains real-time performance at 14 frames per second with only 0.5% compression overhead. Additionally, its query-agnostic nature offers clear benefits in multi-turn dialogue settings (Appendix. C). By eliminating the KV cache bottleneck without retraining or query dependency, InfiniPot-V paves the way for practical, on-device multimodal assistants.

2 Background

We aim to deploy streaming video understanding (SVU) applications [49, 35] in memory-constrained environments. Unlike offline video understanding (OVU) [54, 11, 45], which assumes access to the entire video, SVU must process arbitrarily long video streams and answer questions at any time step using only the frames observed up to that point. Given a video stream $V_T := [v_1, v_2, \ldots, v_T]$ with T frames and a set of questions $Q = q_1, q_2, \ldots, q_N$, SVU answers each question q_i at time t ($1 \leq t \leq T$) using only the observed frames $V_t := [v_1, v_2, \ldots, v_t]$.

As SVU deals with unbounded video streams, memory-efficient processing is essential. In this section, we describe how multimodal large language models (MLLMs) handle long videos, review prior approaches to memory reduction in OVU, and analyze their limitations when applied to SVU. (See Appendix. F for a detailed discussion of related work.)

2.1 Preliminary: Offline Long Video Understanding

Video Processing in MLLMs. Multimodal Large Language Models (MLLMs) [52, 48, 43] process offline videos through a structured pipeline (Fig. 1(a)). Given a video $V_T := [v_1, v_2, \ldots, v_T]$ of T uniformly sampled frames, a vision encoder f_{ViT} transforms each frame into visual tokens:

$$X = f_{\text{ViT}}(V_T) = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{N \times D}, \tag{1}$$

where $N = P \times T$ denotes the total number of sampled tokens, where P is the number of tokens per frame (determined by input resolution and ViT patch size), and D is the token embedding dimension.

The token sequence X is then passed to the LLM in two phases: Prefill and Decoding. During the Prefill phase (Fig. 1(a), step 2), all tokens are processed at once to construct the initial KV cache. The attention operation computes:

$$Q = XW_q, \quad K = XW_k, \quad V = XW_v, \quad O^{\text{attn}} = \text{Softmax}\left(\frac{QK^\top}{\sqrt{D}} + M\right)V, \tag{2}$$

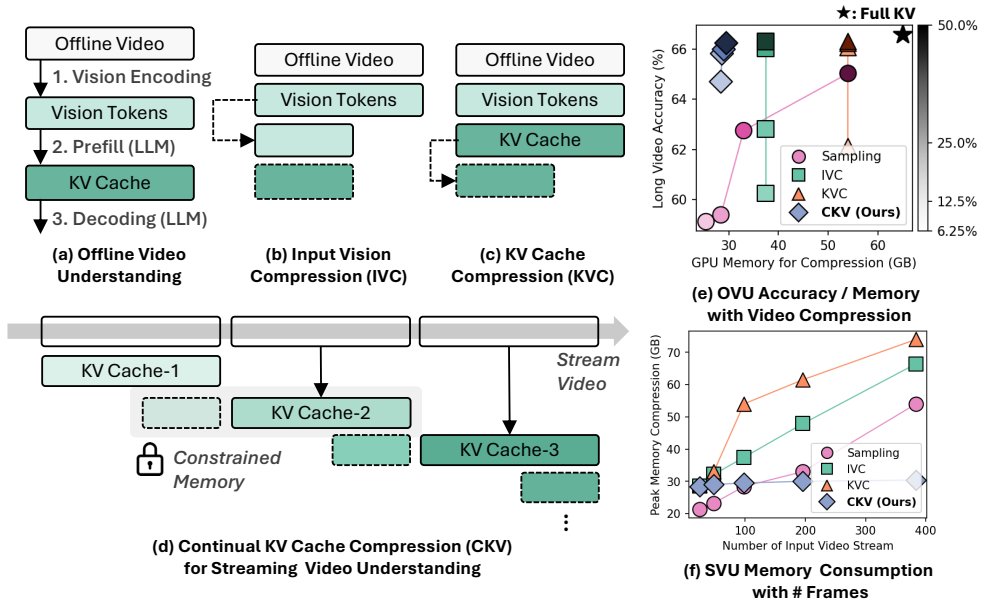


Figure 1: **MLLMs Video Understanding and Compression.** (a) OVU pipeline; (b) IVC: compresses vision tokens after encoding; (c) KVC: compresses KV cache after prefill; (d) CKV: iteratively processes and compresses KV caches to constrain memory usage; (e) Accuracy vs. GPU memory consumption for compression across four token reduction ratios (50%, 25%, 12.5%, 6.25%) on MLVU using `Qwen-2-VL-7B`. LongVU[36] is used for IVC, SnapKV[24] for KVC; (f) GPU memory usage as input video stream length increases. IVC/KVC/CKV target a 6K cache; Sampling uses 1/4 of input frames. Measured on A100 80GB single GPU.

where $W_q, W_k, W_v \in \mathbb{R}^{D\times D}$ are projection matrices and M is a causal mask enforcing autoregressive decoding.

In the Decoding phase (Fig. 1(a), step 3), the model generates tokens one at a time using cached keys and values from the prefill phase. To avoid redundant computation, the KV cache $\mathcal{C} = (K, V)$ is updated incrementally:

$$\mathcal{C}_{t+1} = (\{K, k_{t+1}\}, \{V, v_{t+1}\}), \tag{3}$$

where k_{t+1}, v_{t+1} correspond to the KV embeddings of the newly processed token.

2.2 Offline Long-Video Compression Strategies

Long videos produce extremely long token sequences X, leading to prohibitive GPU memory and latency during decoding. Prior works tackle this bottleneck in the offline setting through three classes of methods (Fig. 1a–c):

(1) Frame Sampling [10]. Uniformly sampling a shorter clip $V'_{T'} \subseteq V_T$ reduces the input length and, hence, memory usage is also reduced proportional to compression rate.

(2) Input-Vision Compression (IVC) [36, 40]. After vision encoding, IVC aggressively prunes redundant vision tokens, keeping only a salient subset $X' \subseteq X$ (Fig. 1b) to shrink the context fed into the language decoder for memory-compressed Prefill.

(3) KV cache Compression (KVC) [24, 12, 4]. Conduct compression after prefill: KVC computes importance scores $u_t = \sum_{i=N-w}^{N} \text{Attn}(x_i \to x_t)$ over the last w tokens and retain top-M entries for the memory budget M by applying eviction policy π, yielding a compressed cache $\mathcal{C}' = \pi(\mathcal{C})$ for memory-compressed Decoding. (Fig. 1c). Note that the π eviction policy is highly dependent on the content of the last w tokens, reflecting the user query, and is thus referred to as query-dependent cache compression method (see Appendix. D for further analysis).

These techniques are effective when the entire video is available upfront, but they implicitly assume (i) unconstrained memory for compression and (ii) a known or easily approximated query.

2.3 Challenges in Streaming Video Understanding

Fig. 1(e) compares three *offline* compression methods on a fixed 50 K-token video at four compression ratios (darker shades indicating higher ratios: 50%, 25%, 12.5%, 6.25%), revealing a fundamental trade-off between memory usage and accuracy. *Frame sampling* skips frames to save memory, but severely degrades recognition accuracy. Increasing the sample ratio improves accuracy but quickly inflates memory usage. *IVC* starts with a large memory footprint for all vision tokens before selecting which to discard. *KVC*, which operates on more expressive key–value features, achieves the highest accuracy but requires the largest Prefill cache. Notably, even under a favorable offline setting—with full video access and an offline query—none of the methods achieve both high accuracy and low memory usage.

This trade-off becomes more severe in the streaming video understanding (SVU) setting. As shown in Fig.1(f), peak GPU memory usage increases with stream length. KVC exhibits near-linear memory growth, as it must materialize all vision tokens and build the full KV cache before compression. Furthermore, due to its query-dependent nature, KVC must *re-execute* the memory-intensive prefill stage whenever the user query changes. Frame sampling and IVC also grow linearly, albeit more slowly, eventually exceeding the memory capacity of practical edge devices (e.g., 32GB[19]) as the stream continues. ReKV [35], a recent KVC method, addresses this by offloading the KV cache to CPU memory, but this introduces substantial offloading overhead and compression latency.

These findings highlight two core requirements for SVU: (1) a fixed memory budget that does not grow with stream length, and (2) query-agnostic token retention strategies. Existing methods fail to meet at least one of these, limiting their suitability for SVU. To overcome this, we propose Continual KV cache compression (CKV), illustrated in Fig. 1(d). CKV processes frames in small blocks and compresses the cache whenever the fixed memory limit is reached, ensuring constant memory usage throughout streaming. Additionally, for query-agnostic token retention, our approach employs lightweight spatiotemporal metrics to identify and preserve semantically significant tokens without relying on future queries. As a result, despite operating under strict memory constraints, CKV achieves accuracy on par with or better than KVC (Fig.1(e)), while consuming far less memory than IVC or frame sampling (Fig. 1(f)). The algorithmic details are described in Sec.3.

Algorithm 1 Continual KV cache Compression (CKV) with InfiniPot-V

Require: Memory budget $|M|$, target cache size $|C|$, TaR ratio α
  Initialize $K, V \leftarrow \emptyset$ ▷ Empty KV cache
  while video stream continues **do**
    (1) Process: $K_{\text{new}}, V_{\text{new}} \leftarrow$ Process new frame; $K \leftarrow [K; K_{\text{new}}]$, $V \leftarrow [V; V_{\text{new}}]$ ▷ Append new tokens
    if len$(K) \geq |M|$ **then** ▷ Memory budget exceeded
      (2) Extract: $K_{\text{recent}}, V_{\text{recent}} \leftarrow$ recent r frames from K, V ▷ len(K) = len$(V) = |M|$
      (3) TaR: $s^{\text{TaR}} \leftarrow$ ComputeTaRScores(K); $\mathcal{I}_{\text{TaR}} \leftarrow$ TopK$(s^{\text{TaR}}, \alpha|C| - \text{len}(K_{\text{recent}}))$ ▷ Sec. 3.1
      (4) VaN: $s^{\text{VaN}} \leftarrow$ ComputeAdaptiveVaNScores(V); $\mathcal{I}_{\text{VaN}} \leftarrow$ TopK$(s^{\text{VaN}}, (1-\alpha)|C|)$ ▷ Sec. 3.2
      (5) Combine: $\mathcal{I} \leftarrow \mathcal{I}_{\text{TaR}} \cup \mathcal{I}_{\text{VaN}} \cup$ Indices(K_{recent}); $K \leftarrow K[\mathcal{I}]$, $V \leftarrow V[\mathcal{I}]$ ▷ Compress to $|C|$ size
    end if
    if user query arrives **then**
      Generate response using current K, V
    end if
  end while

3 InfiniPot-V: Memory-Constrained Streaming Video Understanding

We present **InfiniPot-V**, a CKV framework designed for memory-constrained SVU. As shown in Fig. 1(d) and Algorithm 1, InfiniPot-V processes video streams by applying continual KV cache compression within a fixed memory budget. In this framework, KV embeddings from incoming frames are stored until the memory limit $|M|$ is reached. At that point, compression reduces the cache to a smaller target size $|C|$ ($|M| \gg |C|$), retaining only the most essential vision tokens based on two criteria. The freed space ($|M| - |C|$) accommodates new frames. This process repeats continuously, enabling efficient stream processing under strict memory constraints. When a user query is issued, the model answers using the compressed cache that summarizes visual context from all prior frames. Notably, compression adds only 0.5% overhead relative to input frames processing time.

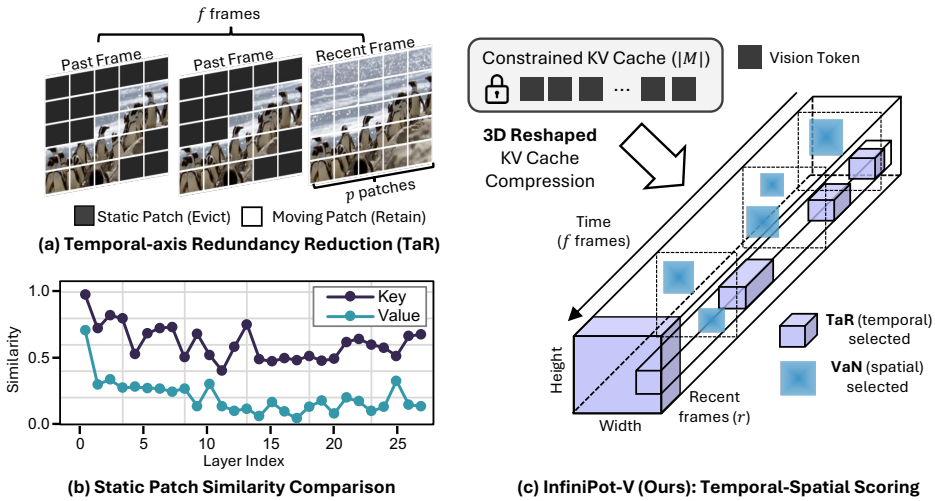


Figure 2: **Spatio-Temporal KV cache Compression (TaR and VaN).** (a) Temporal redundancy across adjacent frames, showing static patches that can be evicted from past frames; (b) Layer-wise cosine similarity of Key/Value embeddings for static patches between consecutive frames in `LLaVA-Next-Video-7B`; (c) InfiniPot-V performs query-agnostic spatiotemporal compression, reducing temporal redundancy with TaR and selecting tokens via VaN spatial scoring.

InfiniPot-V leverages two token eviction criteria: *Temporal-axis Redundancy (TaR)* and *Value Norm (VaN)* for identifying crucial tokens for compressing KV cache. In the following subsections, we detail each criterion, and finally describe how to effectively combine them.

3.1 Temporal-axis Redundancy (TaR) Reduction via Patch-wise Similarity

Video streams exhibit inherent spatiotemporal redundancy across frames [44, 36, 40]. In this section, we focus on exploiting temporal redundancy, as illustrated in Fig. 2(a) where static patches[5] (e.g., background) persist across frames. For MLLMs processing videos with fixed memory usage, identifying this redundancy in KV caches is crucial. Our analysis in Fig. 2(b) reveals that Key embeddings effectively capture temporal redundancy, exhibiting higher cosine similarity for static patches between adjacent frames compared to Value embeddings, across all layers.

Building on this insight, we propose TaR, a technique that performs a patch-wise comparison of Key embeddings along the temporal axis to detect and reduce redundant tokens. As shown in Fig. 2(c), we introduce a 3D reshaping of Key embeddings to enable direct comparison of corresponding patches across frames. Based on this structured KV cache, the TaR implementation starts with a memory constraint of $|M|$ tokens, processing f consecutive video frames, each containing $p = |M|/f$ vision tokens. To maintain temporal continuity, we designate the r latest frames as *recent frames* and retain them in full. The older *past frames* ($f - r$ frames) are selectively compressed based on their patch-wise similarity to recent frames.

To measure the patch-wise similarity between frames, we divide the current Key embeddings $K \in \mathbb{R}^{H\times(f\times p)\times D}$ into $K_{\text{recent}} \in \mathbb{R}^{H\times r\times p\times D}$ and $K_{\text{past}} \in \mathbb{R}^{H\times(f-r)\times p\times D}$, representing the recent and past frames respectively. For each spatial coordinate (i, j), we compute the ℓ_2-normalized cosine similarity between recent and past frames of the same patch coordinate:

$$s^{\text{TaR}}(t, i, j) = -\frac{1}{r}\sum_{t'=1}^{r} \cos\left(K_{\text{past}}^{(t,i,j)}, K_{\text{recent}}^{(t',i,j)}\right). \tag{4}$$

Here, $s^{\text{TaR}}(t, i, j)$ is the importance score of the patch in t-th frame at (i, j) coordinate. The negative sign is applied so that a higher computed score indicates lower redundancy (i.e., the token is more distinctive). This ensures that tokens with less temporal similarity to recent frames are prioritized.

[5]In MLLMs, each vision patch corresponds to a single token, so we use these terms interchangeably.

We then select the least redundant tokens (i.e., higher score) in past frames using the Top-K operator:

$$\mathcal{I}_{\text{TaR}} = \text{TopK}(s^{\text{TaR}}, |C| - |K_{\text{recent}}|), \tag{5}$$

where $|C|$ is the target cache compression size and $|K_{\text{recent}}| = rp$ accounts for the recent frame tokens that are always retained. The compressed key-value pairs are formed by concatenating the selected key frame tokens with all recent frame tokens:

$$\tilde{K}_{\text{TaR}} = \text{Concat}\big(K[:, \mathcal{I}_{\text{TaR}}, :], K_{\text{recent}}\big), \qquad \tilde{V}_{\text{TaR}} = \text{Concat}\big(V[:, \mathcal{I}_{\text{TaR}}, :], V_{\text{recent}}\big). \tag{6}$$

By fully preserving the most recent frames, we maintain complete information on rapidly changing or newly introduced content, while selectively retaining distinctive visual elements from the past.

3.2 Spatial Semantic Importance Preserving with Value Norm (VaN)

While TaR focuses on reducing temporal redundancy, VaN serves a complementary role: identifying and preserving semantically salient regions within each video frame, independent of the query. To achieve this, we employ Value embeddings (V), which inherently capture semantic information in transformer attention [41]. Specifically, we introduce Value Norm (VaN) as a metric for token-level semantic importance: $s^{\text{VaN}} = \|V^{(t,i,j)}\|_2$.

Analysis of Value Norm. We hypothesize that tokens with higher VaN contain richer semantic information, making them more valuable for video understanding. To quantify semantic importance, we project vision token representations from each layer into the vocabulary space [29] and compute the entropy of the resulting word probability distribution, where the higher entropy implies greater informativeness [9, 3]. As shown in Fig. 3 (a), tokens with higher VaN consistently exhibit higher entropy, confirming their semantic significance. This advantage translates to improved performance: Fig. 3 (b) shows that retaining high-VaN tokens achieves substantially higher video understanding accuracy across various compression ratios compared to low-VaN (VaN Reverse) tokens.

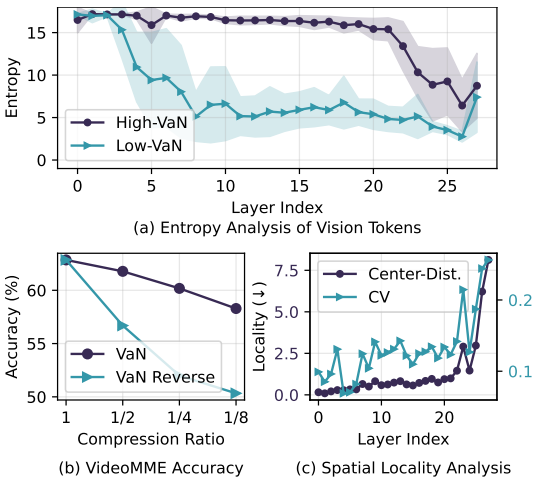


Figure 3: **Value Norm (VaN) Analysis.** (a) Entropy analysis of vision token representations grouped by their VaN scores. (b) VideoMME performance under varying cache compression ratios using either VaN or reverse-VaN for token selection. (c) Layer-wise locality of VaN, measured by center distance and coefficient of variation (CV); lower values indicate stronger spatial consistency. `LLaVA-Next-7B` with Video-MME used.

Layer-wise Adaptive Pooling. An analysis of VaN distributions reveals strong spatial locality patterns in early to middle layers, which gradually diminish in deeper layers as shown in Fig. 3(c). To measure spatial locality patterns across layers, we employ two methods: (1) compute the average distance between the center point and surrounding points within a 3×3 window spanning the VaN values of each frame (center-dist.), and (2) measure the Coefficient of Variance (CV) to quantify dispersion of VaN distributions. Lower values in both metrics—smaller center-dist. and CV—indicate that VaN scores are closely clustered, implying high spatial locality, whereas higher values reflect greater dispersion and lower locality.

As shown in Fig. 3 (c), both metrics consistently indicate strong locality in early to middle layers, while gradually diminishing in deeper layers. Based on this observation, we design an adaptive spatial pooling mechanism that adjusts the average pooling kernel size per layer. To implement this, we design a mapping function g that assigns kernel sizes in inverse relation to each layer's CV:

$$\text{PoolSize}(CV_l) = g(CV_l) \quad \text{where} \quad g : \mathbb{R}^+ \to 1, 3, 5, 7$$

This approach assigns larger pooling kernels (e.g., 7) to lower layers with smaller CV values (higher spatial locality), and smaller kernels (e.g., 1, implying no pooling) to upper layers with larger CV values, thus preserving fine-grained details where needed.

Method Max Duration	**Size**	**# Frames**	**Budget** $\|M\|$	**EgoSchema** 3 min	**MLVU** 120 min	**VideoMME** 60 min	**LVB** 60 min
GPT4-V*	–	1fps	–	55.6	–	60.7	–
GPT4-o*	–	1fps	–	72.2	66.2	77.2	66.7
LLaVA-OV*	7B	32	8K	60.1	64.7	58.2	–
LongVU*	7B	1fps	8K	67.6	65.4	60.6	–
LongVU*	3B	1fps	8K	59.1	55.9	51.5	–
Qwen-2-VL	7B	768	50K	65.2	65.8	63.9	58.8
Qwen-2-VL + **Ours**	7B	768	6K	65.6	65.8	62.8	58.4
LLaVA-Next	7B	128	25K	67.6	68.7	62.8	63.5
LLaVA-Next + **Ours**	7B	128	6K	65.8	65.2	61.1	60.9
Qwen-2.5-VL	3B	768	50K	64.4	63.3	60.3	59.9
Qwen-2.5-VL + **Ours**	3B	768	6K	61.8	62.1	59.3	56.5

Table 1: Comparison of various MLLMs accuracy on four Offline Video Understanding (OVU) benchmarks. * denotes the numbers from official paper.

Compression	**Budget**	**Video MME**			**MLVU**			
Method	$\|M\|$	**Short**	**Med**	**Long**	**Holistic**	**Single**	**Multi**	**Avg.**
FullKV	50K	74.7	62.1	55.0	76.3	73.9	43.3	64.2
TTC [40]	3K	66.8	51.2	47.9	72.1	58.8	33.2	54.8
(IVC)	6K	72.6	55.0	51.7	76.3	60.9	36.7	58.4
STC [36]	3K	67.9	51.0	49.3	71.5	58.6	33.9	55.0
(IVC)	6K	72.6	56.2	51.6	74.3	61.1	35.9	57.9
InfiniPot-V	3K	73.9	57.8	51.8	77.7	70.4	43.2	63.1
(CKV)	6K	74.1	60.8	53.4	77.2	72.3	44.8	64.3

Table 2: Comparison under memory-constrained settings (3, 6K memory-budget) with Input Video Compression (IVC) methods: TTC from DyCoke [40] and STC from LongVU [36]. `Qwen-2-VL-7B` used across VIdeoMME and MLVU benchmarks. Results comparing memory-unconstrained IVC methods (without KV cache compression) with InfiniPot-V are provided in Tab. A6.

For KV cache compression, we select tokens using VaN scores processed through our adaptive pooling mechanism, retaining the Top-$|C|$ tokens with highest pooled VaN values as described in Fig. 2(c): $I_{\text{VaN}} = \text{TopK}(\text{VaN}_{\text{pool}}, |C|)$

$$\tilde{K}_{\text{VaN}} = K[:, \mathcal{I}_{\text{VaN}}, :], \quad \tilde{V}_{\text{VaN}} = V[:, \mathcal{I}_{\text{VaN}}, :]. \tag{7}$$

3.3 Design Space Exploration

Combining TaR and VaN for Token Selection. TaR and VaN capture complementary aspects of spatio-temporal redundancy in streaming video. To integrate them, we prioritize TaR-based selection by first allocating $\alpha|C|$ tokens to TaR, then filling the remaining $(1-\alpha)|C|$ with VaN-selected tokens. This two-stage selection strategy effectively balances temporal and feature importance. A detailed hyperparameter exploration, including sweeps over α and the size of the recent frame window r, is provided in Appendix. A.2.

Comparison with Memory-Constrained Alternatives. A natural question is whether IVC or KVC can be adapted for SVU under memory constraints. To explore this, we apply query-agnostic methods such as spatial token compression (STC) and token temporal merging (TTC) from LongVU [36] and DyCoke [40]. InfiniPot-V outperforms all these baselines by a notable accuracy margin, demonstrating the strength of continual compression over expressive key-value embeddings (details in Tab. 2).

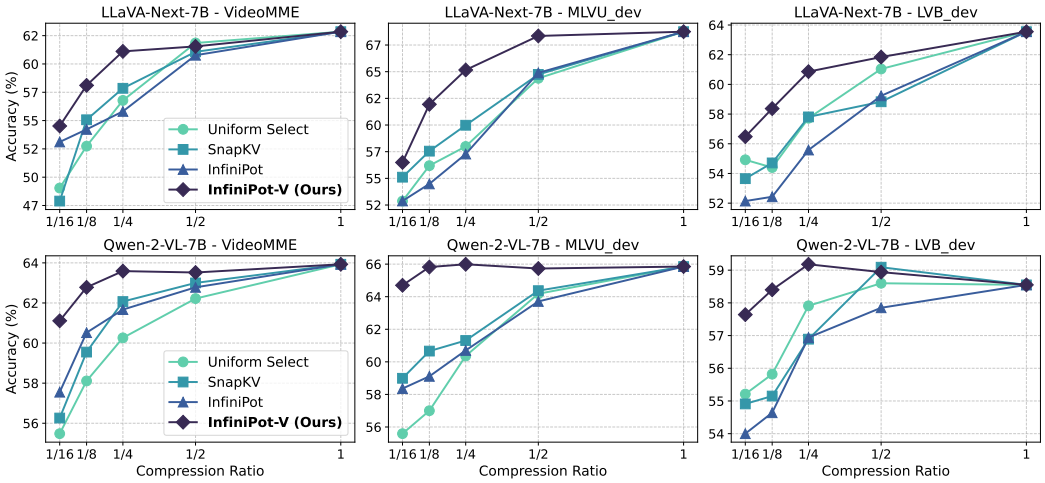


Figure 4: KV cache Compression (KVC) methods evaluation results with offline long video understanding tasks under Continual KV Cache Compression (CKV) framework. Performance across four compression ratios (1/16, 1/8, 1/4, 1/2) for `LLaVA-Next-7B` (top row) and `Qwen-2-VL-7B` (bottom row) on VideoMME, MLVU_{dev}, and $\text{LongVideoBench}_{\text{dev}}$ (LVB_{dev}) tasks. The full evaluation results are shown in Table A5.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate our InfiniPot-V on both offline video understanding (OVU) and streaming video understanding (SVU) tasks. For OVU, we utilize representative multiple-choice based long video understanding benchmarks (ranging from 3 minutes to over 2 hours): VideoMME [11], MLVU [54], LongVideoBench (LVB) [45], and EgoSchema [28].

For SVU, we employ RVS-Ego/Movie streaming video QA benchmark [49], featuring open-ended questions paired with timestamps, and evaluate the answers using `GPT-3.5-turbo-0125` following [49, 35]. We further extend multiple-choice based SVU evaluation, OVO-Bench [23] and StreamingBench [25]. OVO-Bench evaluates temporal reasoning under three scenarios—backward tracing, real-time visual perception, and forward active responding—while StreamingBench evaluates real-time visual comprehension of streaming videos.

Models. We apply our method on four state-of-the-art MLLMs capable of long-video understanding: Qwen-2-VL [43], Qwen-2.5-VL [48], LLaVA-OV-7B [22], and LLaVA-Next-Video [52]. Details on input video sampling settings and benchmark details are provided in Appendix. B.

4.2 Evaluatal Results

Offline Video Understanding. To assess the absolute compression capability of our method, we evaluate InfiniPot-V with both commercial MLLMs (GPT-4V [30], GPT-4o [31]) and state-of-the-art public models designed for offline video understanding, including LLaVA-OV [22], and LongVU [36]. Unlike these specialized, fully trained models, InfiniPot-V is a training-free, plug-in framework compatible with MLLMs of various scales, enabling high performance under fixed memory budgets. As shown in Tab. 1, InfiniPot-V reduces memory usage to just 25% (6K tokens) for LLaVA-Next (originally 25K tokens) and 12.5% for Qwen-VL series (6K vs. 50K), with minimal performance loss. Notably, it achieves comparable or better accuracy than LongVU at the 7B scale and significantly outperforms it at 3B, demonstrating both efficiency and scalability.

Comparison with IVC under Memory Constraints. To evaluate recent query-agnostic IVC methods under memory-constrained CKV, we adopt a unified setup on VideoMME and MLVU: token temporal merging (TTC) from DyCoke [40] and spatial token compression (STC) from LongVU [36] are applied to compress vision tokens to fit the target memory budget $|M|$, while KV cache is

	RVS-Ego		RVS-Movie		Execution Time	Total Memory Usage	
LLaVA-OV-7B	**Acc**	**Score**	**Acc**	**Score**	**Video Enc. (msec/Frame)**	**GPU**	**CPU**
ReKV	60.1	3.9	53.4	3.8	285.7	37.5 GB	+ 18.8GB/h
ReKV w/o off.	55.8	3.3	50.8	3.4	74.6	27.2 GB	0
InfiniPot-V	**57.9**	**3.5**	**51.4**	**3.5**	76.3	27.8 GB	0

Table 3: Streaming benchmark comparison to offloading-based KV cache control method. (ReKV) Video Enc. shows execution time per frame, GPU indicates peak memory usage, and CPU denotes the size of video KV-Cache offloaded to CPU per hour. Results based on an 1-hour video processed with a 0.5 fps sampling rate in streaming mode. `LLaVA-OV-7B is used.`

Qwen-2.5-VL	Scale	OVO-BW	OVO-Real	OVO-FW	OVO-Avg	StreamingBench
Uniform Select	7B	44.5	61.1	47.8	51.7	75.2
InfiniPot-V	7B	47.6	65.9	47.9	**53.6**	**76.4**

Table 4: Comparison on OVO-Bench (BW: Backward, Real: Realtime, FW: Forward) and Streaming-Bench (Real-time visual understanding) using `Qwen-2.5-VL-7B` under 4K memory budget.

managed using a sliding window attention (SWA) [2]. When operated under such constraints, these IVC methods suffer from notable accuracy degradation. In contrast, InfiniPot-V performs KV cache compression using TaR and VaN, leveraging expressive key-value representations to achieve superior average accuracy under a 6K memory budget—corresponding to an 88% lossless compression rate.

Comparison with KVC under Memory Constraints. Fig.4 evaluates KVC methods within our CKV framework under constrained memory across offline video understanding tasks. Compression ratios (1/16, 1/8, 1/4, 1/2) are defined based on each model's maximum frame capacity (e.g., 128 frames for LLaVA-Next, 768 for Qwen-2-VL). Our InfiniPot-V consistently outperforms all baseline methods (Uniform Select, SnapKV, InfiniPot) across all tasks for both LLaVA-Next-7B and Qwen-2-VL-7B, demonstrating superior video understanding performance. Under CKV constraints—where actual query access is not available—query-dependent methods like SnapKV[24] degrade significantly. In contrast, InfiniPot-V maintains strong accuracy even at high compression ratios (e.g., 1/16), thanks to its query-agnostic selection via TaR and VaN.

Streaming Video Understanding. We first evaluate InfiniPot-V on streaming video understanding (SVU) using two StreamingVQA benchmarks, RVS-Ego and RVS-Movie, with LLaVA-OV-7B. As a baseline, we compare against ReKV [35], a state-of-the-art SVU method, under two configurations: (1) CPU–GPU with CPU offloading, which allows spilling KV cache to CPU memory, and (2) CPU–GPU without offloading, simulating shared-memory devices where CPU memory is unavailable or pre-occupied [19]. Tab. 3 reports SVU accuracy, compression time, and memory usage. While CPU offloading enables ReKV to retain the full KV cache, it incurs substantial transfer overhead; without offloading, ReKV suffers from severe accuracy degradation due to limited local memory. In contrast, InfiniPot-V operates entirely within GPU memory, eliminating offloading latency and achieving higher accuracy, making it a practical solution for memory-constrained systems.

To further evaluate accuracy in streaming scenarios, we extend the analysis to two additional benchmarks: OVO-Bench [23] and StreamingBench [25], as summarized in Tab. 4. Compared to the uniform token selection baseline, InfiniPot-V demonstrates consistent gains across all metrics, notably improving recall of past visual information on the OVO-BW (backward) task (44.5 $\rightarrow$ 47.6) and achieving higher real-time understanding accuracy on the OVO-Real (Realtime) (61.1 $\rightarrow$ 65.9) and StreamingBench score (75.2 $\rightarrow$ 76.4), demonstrating the effectiveness of our TaR–VaN compression in eliminating temporal redundancy while preserving spatially salient semantics, showing its capability under challenging SVU settings.

4.3 Ablation Study

Tab. 5 validates our design decisions for TaR and VaN. Reversed strategies (TaR Reverse and VaN Reverse) significantly degrade performance by discarding distinctive or semantically important tokens. Within TaR, patch-wise similarity proves more effective than frame-level similarity (64.5 vs. 62.9).

MLVU_{dev}	Holistic Reasoning		Single Detail			Multi Detail		
Ablation Study	**Topic**	**Anomaly**	**Plot**	**Needle**	**Ego**	**AO**	**AC**	**Avg**
Full KV	85.2	67.5	72.7	83.9	65.1	54.1	32.5	65.9
Uniform Select	83.7	66.5	67.9	76.1	58.5	51.0	27.2	61.5
TaR Reverse	79.0	64.5	56.9	65.6	55.1	45.2	21.8	55.5
TaR Frame	82.9	66.0	67.0	78.9	63.6	51.0	31.1	62.9
TaR	85.9	66.5	71.8	78.0	62.2	51.7	35.4	**64.5**
VaN Reverse	78.3	66.5	56.2	66.8	53.4	46.3	17.5	55.0
VaN	84.4	68.0	68.6	76.6	61.9	52.5	29.1	63.0
VaN + Pool	85.2	68.0	71.4	77.5	63.1	52.1	31.5	**64.1**
TaR + VaN + Pool	86.3	68.0	72.7	80.3	63.9	54.1	35.4	**65.8**

Table 5: Ablation study of TaR, VaN, and their combination. Experiments conducted on MLVU using `Qwen-2-VL-7B` with a 6K memory budget.

Method	**Length (sec)**	**Context Length**	**VRAM (GB) ↓**	**FPS (frame/sec) ↑**	**Throughput (tok/sec) ↑**	**Power (J) ↓**
Full KV	100	18K	13.8	6.2	5.0	1.6
InfiniPot-V			9.2 (1.5×)	6.4 (1.0×)	9.2 (1.8×)	1.2 (1.4×)
Full KV	300	54K	26.6	5.0	2.0	4.7
InfiniPot-V			10.1 (2.6×)	6.4 (1.3×)	9.2 (4.6×)	2.4 (2.0×)
Full KV	500	90K	39.0	4.1	1.2	7.5
InfiniPot-V			10.7 (3.6×)	6.3 (1.5×)	9.1 (7.3×)	3.7 (2.0×)
Full KV	600	108K	OOM	OOM	OOM	OOM
InfiniPot-V			11.3	6.4	9.2	4.3

Table 6: Streaming video processing performance results on NVIDIA Jetson AGX Orin. Streaming 10-minute video samples (FPS 0.2–1) processed with Qwen-2.5-VL-3B, measuring memory, FPS (frames processed per second), throughput, and power consumption.

VaN alone surpasses the baseline, and its performance improves further with adaptive pooling (64.1 vs. 63.0). Combining TaR and VaN yields the highest accuracy, significantly outperforming the baseline. Additional integration explorations are discussed in Appendix. A.2.

4.4 Edge Device Deployment Results

We evaluate our CKV framework—combining continual KV compression with TaR and VaN scoring—on the NVIDIA Jetson AGX Orin [19] using `Qwen-2.5-VL-3B` and 10-minute StreamingBench [25] videos (0.2–1 FPS). As shown in Tab. 6, our method achieves consistent efficiency across all metrics. Peak memory remains nearly constant (9.2–10.7 GB) while Full KV grows linearly (13.8→39.0 GB), yielding a 3.6× reduction at 500 seconds streaming video. Prefill speed, measured in FPS (frames processed per second during prefill), stays stable at 6.3–6.4 FPS, whereas Full KV drops to 4.1 FPS. Generation throughput also increases up to 7.3× (1.2→9.1 tok/sec) under a fixed memory budget. Power usage scales proportionally with compute, improving energy efficiency by nearly 2× (7.5→3.7 J). Notably, CKV enables continuous inference on 600-second streams where Full KV fails with OOM, confirming its practicality for real-time multimodal reasoning on memory-constrained edge devices.

5 Conclusion

In this paper, we proposed InfiniPot-V, a training-free KV cache control framework for streaming video processing in memory-constrained environments. Built around practical constraints—unavailable queries and strict memory budgets during compression—InfiniPot-V employs two novel token eviction criteria, TaR and VaN, achieving significant improvements in long video understanding under streaming scenarios.

References

[1] Kazi Hasan Ibn Arif, JinYi Yoon, Dimitrios S. Nikolopoulos, Hans Vandierendonck, Deepu John, and Bo Ji. Hired: Attention-guided token dropping for efficient inference of high-resolution vision-language models, 2024.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[3] David Chan, Rodolfo Corona, Joonyong Park, Cheol Jun Cho, Yutong Bai, and Trevor Darrell. Analyzing the language of visual tokens, 2025.

[4] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models, 2024.

[5] Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, Ethan He, Hongxu Yin, Pavlo Molchanov, Jan Kautz, Linxi Fan, Yuke Zhu, Yao Lu, and Song Han. LongVILA: Scaling long-context visual language models for long videos. In *The Thirteenth International Conference on Learning Representations*, 2025.

[6] Giulio Corallo, Orion Weller, Fabio Petroni, and Paolo Papotti. Beyond rag: Task-aware kv cache compression for comprehensive knowledge reasoning, 2025.

[7] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[8] Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for KV cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18476–18499, November 2024.

[9] Sebastian Farquhar, Jannik Kossen, Livia Kuhn, et al. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024.

[10] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2019.

[11] Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024.

[12] Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Google DeepMind. Project ASTRA. `https://deepmind.google/technologies/project-astra/`, 2024.

[15] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina González, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jáchym Kolář, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano,

Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbeláez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18995–19012, June 2022.

[16] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, June Paik, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed attention: Accelerating long context length llm inference. *arXiv preprint arXiv:2411.09688*, 2024.

[17] Qingqiu Huang, Yu Xiong, Anyi Rao, Jiaze Wang, and Dahua Lin. Movienet: A holistic dataset for movie understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 709–727. Springer, 2020.

[18] Yuxiang Huang, Binhang Yuan, Xu Han, Chaojun Xiao, and Zhiyuan Liu. Locret: Enhancing eviction in long-context llm inference with trained retaining heads on consumer-grade devices, 2025.

[19] Leela S. Karumbunathan. NVIDIA Jetson AGX Orin Series Technical Brief. Technical Report TB_10749-001_v1.2, NVIDIA Corporation, July 2022. Version 1.2.

[20] Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. Kvzip: Query-agnostic kv cache compression with context reconstruction, 2025.

[21] Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot: Infinite context processing on memory-constrained LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16046–16060, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[22] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

[23] Yifei Li, Junbo Niu, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. Ovo-bench: How far is your video-llms from real-world online video understanding?, 2025.

[24] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[25] Junming Lin, Zheng Fang, Chi Chen, Zihao Wan, Fuwen Luo, Peng Li, Yang Liu, and Maosong Sun. Streamingbench: Assessing the gap for mllms to achieve streaming video understanding. *arXiv preprint arXiv:2411.03628*, 2024.

[26] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ringattention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024.

[27] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc., 2023.

[28] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[29] Clement Neo, Luke Ong, Philip Torr, Mor Geva, David Krueger, and Fazl Barez. Towards interpreting visual information processing in vision-language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[30] OpenAI. Gpt-4v(ision) system card, 2023.

[31] OpenAI. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`, 2024.

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

[33] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[34] Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. Introducing gemini 2.0: our new ai model for the agentic era. `https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024`, 2024.

[35] Zhelun Yu Shangzhe Di. Streaming video question-answering with in-context video KV-cache retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025.

[36] Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, Zhuang Liu, Hu Xu, Hyunwoo J. Kim, Bilge Soran, Raghuraman Krishnamoorthi, Mohamed Elhoseiny, and Vikas Chandra. Longvu: Spatiotemporal adaptive compression for long video-language understanding, 2024.

[37] Morgan Stanley. Humanoids: Investment implications of embodied ai. Technical report, Morgan Stanley, June 2024. Accessed via Future Management Group: https://www.futuremanagementgroup.com/wp-content/uploads/240626-Humanoid-Robots-Morgan-Stanley.pdf.

[38] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[39] Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Danning Ke, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient KV cache compression through retrieval heads. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models, 2024.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[42] Ethan Waisberg, Joshua Ong, Mouayad Masalkhi, Nasif Zaman, Prithul Sarker, Andrew G Lee, and Alireza Tavakkoli. Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments. *Eye*, 38(6):1036–1038, 2024.

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

[44] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[45] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[46] Mingze Xu, Mingfei Gao, Shiyu Li, Jiasen Lu, Zhe Gan, Zhengfeng Lai, Meng Cao, Kai Kang, Yinfei Yang, and Afshin Dehghan. Slowfast-llava-1.5: A family of token-efficient video large language models for long-form video understanding, 2025.

[47] Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[48] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

[50] Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster, 2024.

[51] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. Sparsevlm: Visual token sparsification for efficient vision-language model inference, 2024.

[52] Yuanhan Zhang, Bo Li, haotian Liu, Yong jae Lee, Liangke Gui, Di Fu, Jiashi Feng, Ziwei Liu, and Chunyuan Li. Llava-next: A strong zero-shot video understanding model, April 2024.

[53] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[54] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024.

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:

   Guidelines:

   - The answer NA 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 NA 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:

   Guidelines:

   - The answer NA 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: [NA]

Justification:

Guidelines:

- The answer NA 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:

   Guidelines:

   - The answer NA 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: The code and data involve proprietary components that are subject to company policy restrictions, and thus cannot be publicly released. However, detailed implementation settings and reproduction instructions are thoroughly documented in the main text and Appendix A and B.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA 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: [Yes]

   Justification:

   Guidelines:

   - The answer NA 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: [Yes]

   Justification:

   Guidelines:

   - The answer NA 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:

   Guidelines:

   - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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: [NA]

    Justification:

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Algorithm 2 InfiniPot-V Algorithm

Require: Video stream V, memory constraint $|M|$, target KV cache size $|C|$, recent frame count r, CV thresholds $\{\tau_1, \tau_2, \tau_3\}$, TaR ratio $\alpha \in [0, 1]$, f frames corresponding to $|M|$ tokens, vision token number per single frame $p = |M|/f$

Ensure: Compressed KV cache $\{\tilde{K}_l, \tilde{V}_l\}_{l=1}^{L}$

1: Let $C_{\text{TaR}} = \alpha|C|$ be the TaR selection budget
2: Let $C_{\text{VaN}} = (1 - \alpha)|C|$ be the VaN selection budget
3: Initialize empty KV cache for each layer $l \in \{1, \ldots, L\}$
4: **while** processing video stream V **do**
5: Accumulate KV embeddings until reaching $|M|$
6: **for** each layer l **do**
7: // Temporal-axis Redundancy (TaR)
8: Reshape K_l into $K_{\text{recent},l} \in \mathbb{R}^{H \times r \times p \times D}$ and $K_{\text{past},l} \in \mathbb{R}^{H \times (f-r) \times p \times D}$
9: **for** each patch (t, i, j) in past frames **do**
10: $s(t, i, j) = -\frac{1}{r} \sum_{t'=1}^{r} \cos(K_{\text{past},l}^{(t,i,j)}, K_{\text{recent},l}^{(t',i,j)})$
11: **end for**
12: $\mathcal{I}_l \leftarrow \text{TopK}(S_l, C_{\text{TaR}})$ ▷ Select least redundant tokens
13: // Value Norm (VaN) with Adaptive Pooling
14: $\text{VaN}_l \leftarrow \|V_l\|_2$
15: // Compute CV for adaptive pooling
16: $\mu_l \leftarrow \text{mean}(\text{VaN}_l)$
17: $\sigma_l \leftarrow \text{std}(\text{VaN}_l)$
18: $\text{CV}_l \leftarrow \sigma_l / \mu_l$
19: // Determine pooling size using mapping function g
20: $\text{pool\_size}_l \leftarrow g(\text{CV}_l)$ ▷ Using thresholds $\{\tau_1, \tau_2, \tau_3\}$
21: **where** $g(\text{CV}) = \begin{cases} 7, & \text{if CV} < \tau_1 \\ 5, & \text{if } \tau_1 \leq \text{CV} < \tau_2 \\ 3, & \text{if } \tau_2 \leq \text{CV} < \tau_3 \\ 1, & \text{if CV} \geq \tau_3 \end{cases}$
22: $\text{VaN}_{pool,l} \leftarrow \text{AveragePool2d}(\text{VaN}_l, \text{pool\_size}_l)$
23: // Combine TaR and VaN by prioritizing TaR-selected tokens
24: $\text{VaN}_{pool,l}[\mathcal{I}_l] \leftarrow \max(\text{VaN}_{pool,l})$ ▷ Prioritize TaR tokens
25: $\mathcal{J}_l \leftarrow \text{TopK}(\text{VaN}_{pool,l}, |C|)$ ▷ Final token selection
26: $\tilde{K}_l \leftarrow K_l[:, \mathcal{J}_l, :]$, $\tilde{V}_l \leftarrow V_l[:, \mathcal{J}_l, :]$ ▷ Update layer KV cache with compressed KV cache
27: **end for**
28: **end while**

A InfiniPot-V Algorithm and Configuration

A.1 Algorithm Description

Algorithm 2 presents the complete process of InfiniPot-V's cache control framework along with its compression formulation. InfiniPot-V processes video streams by continuously pre-filling and compressing the KV cache using two token selection strategies: Temporal-axis Redundancy (TaR) and Value Norm (VaN). For TaR, the algorithm splits video frames into recent frames (the latest r frames) and past frames, then computes cosine similarities between corresponding patches to identify and remove redundant visual tokens. (Line 10)

For spatial semantic importance token selection, a layer-wise adaptive pooling mechanism based on VaN is employed. The pooling size is dynamically determined by the Coefficient of Variation (CV) of the VaN, (Line 18) where a higher CV indicates a sparser or more distinct feature distribution. Pre-computed model-specific CV thresholds $\{\tau_1, \tau_2, \tau_3\}$ determine pooling sizes from the set {1,3,5,7}, selecting larger windows for uniform (low CV) VaN distributions and smaller ones for sparse (high CV) VaN distributions (Line 21).

To integrate both criteria, TaR-selected tokens are prioritized by assigning them the maximum VaN score before the final token selection. Specifically, by setting $\text{VaN}_{pool,l}[\mathcal{I}_l] = \max(\text{VaN}_{pool,l})$ (Line 24) and then applying a TopK selection, the algorithm ensures that temporally distinctive tokens are preserved while allowing VaN to select additional tokens based on semantic importance.

VideoMME	$\lvert M\rvert = \alpha\lvert\text{TaR}\rvert + (1-\alpha)\lvert\text{VaN}\rvert$, $\lvert M\rvert = 6\text{K}$					
Qwen-2-VL-7B	$\alpha = 0$ (VaN)	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1$ (TaR)
Short (-3 min)	74.4	74.3	74.1	74.9	74.4	73.8
Medium (3 - 30 min)	59.9	59.3	61.3	61.4	61.2	58.2
Long (30 - 120 min)	51.9	51.0	52.4	53.1	53.4	53.2
Average	62.1	61.6	62.6	**63.1**	<u>63.0</u>	61.7
LLaVA-Next-7B	$\alpha = 0$ (VaN)	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1$ (TaR)
Short (-3 min)	69.8	69.8	72.0	71.1	71.9	68.8
Medium (3 - 30 min)	59.3	59.3	59.2	58.7	57.7	57.3
Long (30 - 120 min)	52.1	52.1	52.7	52.0	51.4	51.0
Average	60.4	60.4	**61.3**	<u>60.6</u>	60.3	59.0

Table A1: **TaR and VaN Combination Ratio**: Sweep over combination ratio α in TaR and VaN combination under a 6K memory budget ($|M|$) on VideoMME. α=0 and α=1 correspond to VaN-only and TaR-only, respectively. The best-performing configurations are shown in **bold**, while the second-best results are <u>underlined</u>.

Qwen-2-VL-7B	MLVU				VideoMME			
$\lvert M\rvert = 6\text{K}$	Holistic	Single	Multi	Avg.	Short	Medium	Long	Avg.
$r = f \times 0.125$	77.6	66.2	43.9	<u>63.1</u>	68.7	57.3	51.0	**59.0**
$r = f \times 0.25$	77.8	67.1	43.5	**63.4**	68.0	57.2	51.1	<u>58.8</u>
$r = f \times 0.50$	78.6	64.6	39.0	61.3	65.1	56.7	49.7	57.2
$\lvert M\rvert/\lvert C\rvert = 0.75$	78.2	71.7	44.6	**65.8**	74.4	59.9	51.9	<u>62.1</u>
$\lvert M\rvert/\lvert C\rvert = 0.50$	76.8	68.2	42.3	<u>63.3</u>	74.1	60.3	52.9	**62.4**
$\lvert M\rvert/\lvert C\rvert = 0.25$	73.2	65.9	39.1	60.3	73.6	56.9	50.4	60.3

Table A2: **Recent Frame and Compression Ratio Exploration**: **Top**: Sweep over recent frame numbers r determined by multiplying various ratios $(0.125, 0.25, 0.5)$ to f (frame number corresponding to memory budget $|M|$) in TaR. **Bottom**: Performance under varying compression ratios $|M|/|C|$ across MLVU and VideoMME with `Qwen-2-VL-7B`. TaR performs best with $r \leq 0.25$ and compression ratio ≥ 0.5. The highest values are shown in **bold**, with the second-highest <u>underlined</u>.

A.2 Hyper-Parameter Exploration

InfiniPot-V involves three main hyper-parameters: the TaR and VaN budget allocation ratio α, the number of recent frames r used in TaR, and the target compression size C applied at each continual KV cache compression step. This section presents comparative experiments exploring each hyper-parameter.

TaR and VaN Budget Ratio (α) We compare the accuracy of offline video understanding (OVU) task across different values of α, which determines the budget allocation between TaR and VaN under a fixed memory budget ($|M| = 6\text{K}$), for both Qwen-2-VL-7B and LLaVA-Next-7B models. As shown in Tab. A1, performance peaks when α is between 0.4 and 0.6, outperforming the use of either VaN-only ($\alpha = 0$) or TaR-only ($\alpha = 1$). This confirms the effectiveness of our approach, which jointly considers both spatial and temporal dimensions for KV cache compression.

Recent Frames (r) and Compression Ratio ($|M|/|C|$) Tab. A2 presents exploration experiments for two key hyperparameters: the recent frames number r, which determines the proportion of recent frames within the memory budget in TaR, and the compression ratio $|M|/|C|$, which defines what proportion of the compression size $|C|$ to maintain relative to the memory budget $|M|$ in continual KV cache compression (CKV).

For the recent frame number r (Tab. A2, Top), we observe optimal performance on both MLVU and VideoMME benchmarks when $r \leq 0.25f$. Setting $r = 0.5f$ results in an excessive number of frames being designated as the latest frames for temporal redundancy measurement, which limits the effectiveness of redundancy reduction. This limitation is reflected in the decreased performance

metrics. (61.3 vs 63.4 in MLVU and 57.2 vs 59.0 in VideoMME) Note that the r sweep experiments are conducted using TaR-only settings ($\alpha = 1$).

For the compression ratio ($|M|/|C|$), we conduct comparative experiments across three ratios (0.75, 0.50, and 0.25). As shown in Tab. A2 Bottom, an excessive compression ratio such as 0.25 in CKV results in noticeable performance degradation. These findings confirm that a ratio of 0.5 or higher represents an appropriate configuration for CKV.

Based on these explorations, we standardize the hyperparameter values at $\alpha = 0.5$, $r = 0.125$, and $|M|/|C| = 0.75$ for all main experimental results when evaluating InfiniPot-V.

B Experimental Setting Details

B.1 MLLMs Video Sampling Details.

For all benchmarks, we employ a consistent uniform frame sampling strategy to ensure maximized long video understanding performance across all settings. For Qwen-2-VL [43], which supports dynamic image resizing based on the number of frames, we use the hyper-parameter configuration reported to yield the best performance in their original work: `FPS_MAX_FRAMES` = 768, `VIDEO_MIN_PIXEL` = $128 \times 28 \times 28$ and `VIDEO_MAX_PIXEL` = $768 \times 28 \times 28$. Although theoretically larger token budgets could be set, we adopt this configuration to match the optimal context length of 50K as reported in the original paper [43], on top of which we applied KV cache compression. For LLaVA, we set the number of sampled frames to 128 to ensure it remained within the model's trained context length (<32K). With this video sampling configuration, Qwen-2-VL [43] uses 384 frames with 130 tokens per frame, resulting in a total context length of 49,920 tokens, while LLaVA-Next [52] and LLaVA-OV [22] use 128 frames with 196 tokens per frame, yielding a total of 25,088 for offline long video inputs.

B.2 Long Video Understanding Benchmark Details

Offline Video Understanding(OVU) We evaluate our method on four multiple-choice based offline video question answering benchmarks: Video-MME [11], MLVU [54], EgoSchema [28], and LongVideoBench [45]. For MLVU and EgoSchema, we use the development sets for evaluation.

For Video-MME, we report results without subtitles version. This is because prepending subtitles for all video frames as a single context block directly before the question represents an unrealistic setting that is incompatible with streaming scenarios, where subtitles are typically unavailable during real-time video processing and would not be accessible as complete context in advance.

Streaming Video Understanding (SVU) For SVU evaluation, we use two benchmarks: RVS-Ego and EVS-Movie [49]. RVS-Ego is constructed from 10 videos from the Ego4D [15] dataset, while RVS-Movie uses 22 long videos from MovieNet [17]. Each benchmark consists of a QA set containing open-ended generation questions and their corresponding timestamps indicating when each question should be presented during video streaming.

The evaluation process works as follows: during CKV processing, when the video stream reaches the timestamp of a given question sample, we present the question and generate an answer based on the compressed KV cache accumulated up to that point. The generated answers are then compared against ground-truth answers using `GPT-3.5-turbo-0125` to produce accuracy and score metrics.

B.3 Baseline Settings

Input Video Compression (IVC) Details. For the comparison with Input Video Compression (IVC) methods in Tab. 2 and A6, we implement LongVU [36] and DyCoke [40] as follows: For **LongVU**, we apply Spatial Token Compression (STC) every 8 frames as specified in the original paper. STC compresses vision token embeddings by identifying temporally redundant patches using cosine similarity between patches. We adjust the similarity threshold to control the compression rate while maintaining the original methodology. For **DyCoke**, we implement Token Temporal Merging (TTM) which, similar to LongVU, compresses vision encoder output features. TTM calculates cosine similarity between patches in adjacent frames to eliminate redundant patch embeddings. Following

the original paper, we process compression every 4 frames and adjust the similarity threshold to control compression rates.

For fair comparison in the Continual KV cache compression (CKV) framework in Tab. 2, we adapt both methods to work within memory constraint $|M|$. Specifically, we compress each input video stream to size $|C|$ and implement sliding window attention [2] to evict older KV cache entries once the cache size reaches the predefined memory limit ($|M|$). This adaptation ensures all methods operate under identical memory constraints for a fair comparison with InfiniPot-V. For benchmark comparing InfiniPot-V with IVC methods that use full vision encoding without cache compression, see Tab. A6.

KV Cache Compression (KVC) Details. In Fig. 2, we compare three KV cache compression methods within Continual KV cache compression (CKV). First, **Uniform Select**, inspired by uniform video sampling approaches, selects frames at regular intervals and retains all KV cache tokens corresponding to those frames. For **SnapKV** [24], we follow the original method configuration under the CKV process, using the last 32 tokens of the given budget $|M|$ tokens as an observation window (w) to calculate attention scores for token selection (see Eq. 1 in Appendix. D.1). Additionally, we apply 1D pooling with a kernel size of 7 to these scores, as done in the original implementation[6]. For **InfiniPot** [21], we design a proxy prompt for video compression: "Provide a detailed description of this video." This prompt is utilized in the CaP method to generate attention scores and apply KV cache compression. Detailed experimental results are provided in Tab. A5.

FastV Hyper-Parameter Settings. To provide additional performance comparison with compression methods specialized for MLLMs, we also include performance comparisons with FastV [4] in Tab.A5. FastV requires two hyper-parameters, L and R, which specify the layer where token pruning begins and the percentage of tokens to prune. For a fair comparison, we adjust the R of FastV to ensure that the total number of KV cache entries across layers matches the total entry count of other baselines that maintain the same number of KV-cache entries across each layer. Specifically, for Qwen-2-VL, the (L, R) pairs corresponding to memory budgets of 3K, and 6K are set to (2, 2.8%) and (2, 5.8%) respectively.

B.4 Positional Encoding Details.

MLLM backbone LLMs utilize positional encoding to differentiate vision token positions. LLaVA-Next [52] and LLaVA-OV [22] use standard 1D RoPE [38], while Qwen-2-VL [43] employs 3D RoPE for multimodal encoding. For Offline Video Understanding(OVU), we apply KV cache compression after positional encoding (i.e., Post RoPE). However, Streaming Video Understanding(SVU) presents a challenge: continuous video stream processing can exceed the model's maximum positional range. For example, in LLaVA models with 196 tokens per frame, streaming more than 6 minutes of video at 0.5 FPS exceeds the 32K context window (note that RVS-Ego and RVS-Movie average over 60 minutes).

To address this, we adopt strategies from InfiniPot [21] and ReKV [35], re-ordering positional indices to fit within the memory budget $|M|$ at each CKV step. Specifically, we cache the pre-positional encoded KV's hidden states and re-assign positional indices during decoding, ensuring they never exceed $|M|$ position regardless of video length. While this enables SVU for arbitrarily long videos, it discards the original positional information of vision tokens. In particular, additional handling is required for Qwen-2-VL's 3D RoPE. Developing methods that preserve the original spatial and temporal position encoding while supporting streaming video lengths beyond the model's positional capacity remains an open direction for future work.

C Multi-Turn Video Understanding Analysis

Fig. A1 presents a qualitative comparison between query-dependent (SnapKV)[24] and query-agnostic (InfiniPot-V) KV cache compression approaches in multi-turn conversations with streaming video input. When SnapKV performs compression based on Q1, it generates answers almost identical to the Full KV cache for that specific query (Q1), answering that the butter was placed in the refrigerator.

[6] https://github.com/FasterDecoding/SnapKV

Q1: Where did I put the butter after eating? [Single-Detail]
Full-KV: After eating, the person put the butter in the **refrigerator.** SnapKV: The person put the butter in the **fridge** after eating. **InfiniPot-V**: The person in the video put the butter in the **fridge** after eating.
Q2: When turning off the stove, what kind of kitchen utensil were on the stove? [Single-Detail]
Full-KV: There was a **kettle** on the stove. SnapKV: There were **two spoons** on the stove. ✗ **InfiniPot-V**: There was a **kettle** on the stove.
Q3. How many pieces of bread appear in total during the cooking process? [Multi-Detail]
Full-KV: The video shows that there are **two pieces** of bread in the pan and **two pieces** of bread on the plate, total of **4 pieces of bread.** SnapKV: The video shows that there are **two pieces** of bread in total during the cooking process. ✗ **InfiniPot-V:** The video shows a total of **4 pieces** of bread being used during the cooking process.

Figure A1: **Qualitative Results of Multi-Turn Conversation**: Full-KV uses 16K cache while InfiniPot-V and SnapKV employ 3K compressed KV cache. SnapKV performs query-guided cache compression based on Q1 before proceeding with multi-turn conversation. The video sample is from the MLVU ego reasoning subtask, using the `Qwen-2-VL-7B model`. 128 frame sampling is used.

However, this query-guided compression strategy reveals significant limitations when handling different types of queries (Q2, Q3) about the same video content. Specifically, SnapKV makes critical errors in subsequent queries - misidentifying a kettle as "two spoons" in Q2 and incorrectly counting the total number of bread pieces in Q3.

In contrast, InfiniPot-V maintains accurate answers consistently across all three queries using the same 3K compressed KV cache. It correctly identifies that the butter was placed in the fridge (Q1), recognizes the kettle on the stove (Q2), and counts all 4 pieces of bread throughout the cooking process (Q3), demonstrating the effectiveness of query-agnostic compression for multi-turn streaming video scenarios.

D Why Query-Agnostic KV Cache Compression Matters for SVU?

In this section, we provide a detailed analysis of why query-agnostic compression is essential for Streaming Video Understanding (SVU), building upon the requirements discussed in Sec. 2. To demonstrate how these SVU-specific constraints impact existing KV cache compression methods, we present a case study across three representative scenarios.

D.1 Preliminary: Attention-based KV Cache Compression

Eviction-based KV cache compression reduces cache size by removing tokens with the lowest importance scores. Employing attention scores for computing token importance scores is the predominant approach in previous methods [24, 4, 12, 16].

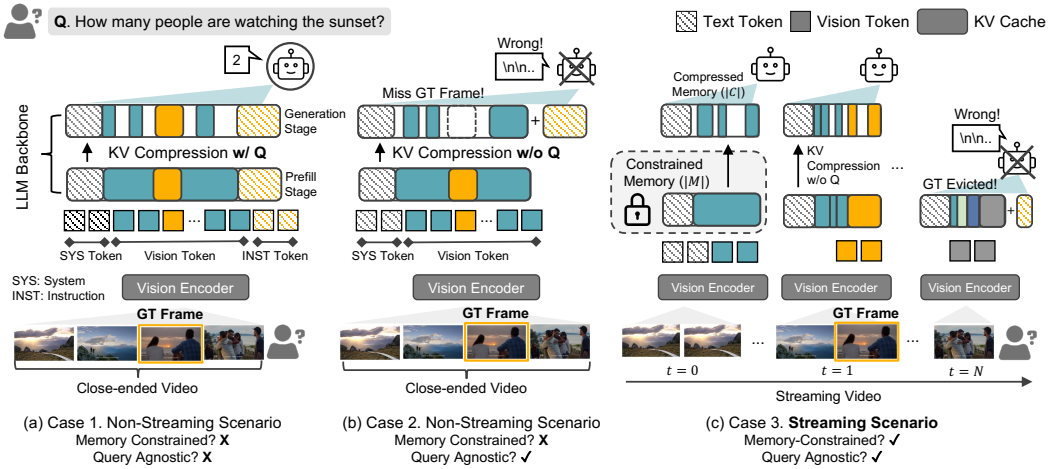
In methods such as SnapKV [24], the importance scores u_t of a token x_t are computed by aggregating attention scores from the last w tokens (i.e., observation window) which contain the user instruction:

$$u_t = \sum_{i=N-w}^{N} \text{Attn}(x_i \rightarrow x_t), \tag{1}$$

where N is the current sequence length. Using these scores, the KV cache is compressed by retaining the top-M tokens with the highest aggregated attention scores. Here, M defines the memory budget: $\mathcal{I} = \text{TopK}(u, M)$ and $u = [u_1, \cdots, u_N]$ indicates the importance scores of all tokens. The compressed Key and Value caches are then formed by extracting tokens at indices $\mathcal{I}$:

$$\tilde{K} = K[:, \mathcal{I}, :], \quad \tilde{V} = V[:, \mathcal{I}, :] \tag{2}$$

where $K, V \in \mathbb{R}^{H \times N \times D}$ are the uncompressed Key and Value caches with H heads, N tokens, and per-head dimension D. This approach has two characteristics: (1) it requires computing the full KV cache for all tokens before compression, and (2) it requires the user query to be present at the end



(a) Case 1. Non-Streaming Scenario
Memory Constrained? ✗
Query Agnostic? ✗

(b) Case 2. Non-Streaming Scenario
Memory Constrained? ✗
Query Agnostic? ✓

(c) Case 3. **Streaming Scenario**
Memory-Constrained? ✓
Query Agnostic? ✓

Figure A2: **KV Cache Compression Case Study with SVU**: Illustration of cache control strategies under three conditions, differing in the presence of two core requirements for Streaming Video Understanding (SVU): memory constrained (MC) and query agnostic (QA). (a) Case 1: Query-guided compression retains relevant (GT) frames for accurate responses. (b) Case 2: Without query guidance, compression fails to preserve critical frames, resulting in inaccurate responses. (c) **Case 3 (Streaming scenario)**: In streaming video processing, where frames arrive continuously, continual KV cache compression (CKV) is necessary, but queries are unavailable during compression.

of the context. We refer to these approaches as *query-guided* or *attention-based* cache compression methods.[7]

D.2 Case Study: Towards Streaming Video Understanding with CKV

To investigate the applicability of attention-based KV cache compression methods to streaming video understanding, we examine three cache control strategies (Fig. A2).

Case 1. Recent KV cache compression methods [24, 12] assume full access to context and queries at compression time, as shown in Fig. A2(a). In this memory-unconstrained setting, the model observes the full input before compression. Previous works [24] have demonstrated that attention scores effectively identify query-relevant tokens KV cache (orange box corresponding to GT Frame in Fig. A2(a)), enabling compression that retains critical information while discarding less important tokens. As shown in Tab. A3, this approach maintains performance comparable to the uncompressed cache setup (68.01 vs 68.75) at the cost of large memory usage at compression, detailed in Fig. 1.

Case	x_t Attention Scoring	Prefill $\lvert M\rvert$	Gen. $\lvert M\rvert$ = 3K	Gen. $\lvert M\rvert$ = 6K
Full KV	n/a	25K	68.75 (↑)	
Case 1	$\text{Attn}(q \to x_t)$	25K	68.01	68.40
Case 2	$\text{Attn}(q' \to x_t)$	25K	60.35	63.42
	$\text{Attn}(q'' \to x_t)$		60.60	63.50
	$\text{Attn}(q_v \to x_t)$		60.32	62.28
Case 3	$\text{Attn}(q_v \to x_t)$	3K/6K	57.55	59.98

Table A3: **Case study of Attention Scoring**: conducted on MLVU benchmark with `LLaVA-Next-Video-7B`. Note that memory-constrained setting (Case 3) shares the same budget during prefill and generation stages.

Case 2. Fig. A2(b) illustrates how attention-based cache compression fails when user queries are unavailable during compression. Under this scenario, although the memory budget is assumed unconstrained, the KV cache is compressed without consideration of (future) queries, causing important visual tokens (orange tokens cache corresponding to the GT Frame) lost during compression. To quantify this degradation and explore alternatives, we test compression with generic queries (q': "What is happening in this video?", q'' :"What are the key events in this video?") and the last vision tokens (q_v) for importance scoring:

[7]Throughout this paper, "query" refers to the user's instruction or question related to the given video.

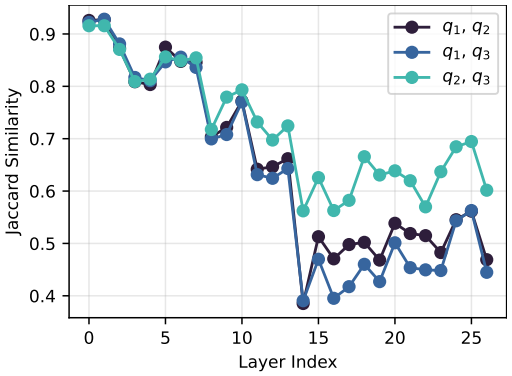


Figure A3: **Jaccard Similarity between KV Caches**: Compare KV cache sets selected by different queries (q_1, q_2, q_3) across layers.

Context Length	Case 1, 2		Case 3 (Ours)	
	Mem (GB)	TTFT (s)	Mem (GB)	TTFT (s)
5K	21.29	0.98	20.93	1.08
25K	33.76	1.21	21.60	1.12
50K	58.55	2.12	22.16	1.17
100K	79.38	3.27	22.85	1.20

Table A4: **Peak GPU Memory and TTFT**: Comparison of peak memory usage and Time-To-First-Token(TTFT) across different context lengths for memory-unconstrained (Case 1, 2) and memory-constrained (Case 3) approaches.

$$u_t^{\text{alt}} = \text{Attn}(q_{\text{alt}} \to x_t), \quad q_{\text{alt}} \in \{q', q'', q_v\} \tag{3}$$

Tab. A3 and Fig. A2(a) show that these alternatives significantly degrade performance (60.32 vs 68.75), even with unconstrained memory.

Case 3: Streaming Scenario. Beyond the query-agnostic challenge in Case 2, deploying streaming video understanding on resource-constrained devices requires fixed memory usage for the KV cache. For input video streams, these constraints necessitate continual compression when new frames arrive and memory capacity is reached, as shown in Fig. A2(c). To evaluate this scenario, we use the query-agnostic approach from Case 2 with vision tokens (q_v) for importance scoring, while compressing the KV cache whenever memory limits are reached. As shown in Tab. A3, this combined constraint further degrades performance (57.55 vs 60.32), highlighting the challenge of preserving key information under both query-agnostic and memory-constrained settings.

This case study reveals two key challenges for KV cache compression in streaming video: 1) the need for *query-agnostic compression* due to continuous incoming video, and 2) the requirement to maintain *fixed memory constraints*. These challenges cause significant performance drops in previous methods [24, 21, 4], motivating Continual KV cache compression (CKV) specifically designed for memory-constrained streaming video.

Attention Scoring Analysis We further analyze the query-dependent nature of attention-based KV cache compression using the VideoMME benchmark. To investigate why performance varies with different queries, we compute the Jaccard similarity between token sets selected for different queries across layers using attention scores at each layer. For this analysis, q_1, q_2, q_3 represent three distinct questions associated with the same video sample in the VideoMME benchmark. As shown in Fig. A3, the similarity between token sets decreases significantly in the middle-to-late layers, dropping to around 0.4. This indicates that each query selects a different set of tokens, particularly in deeper layers. This analysis highlights that attention-based scoring methods inherently select query-specific tokens, explaining the performance degradation when query information is unavailable or changes during streaming video scenarios.

E Memory and Latency Measurement Results

Table A4 presents measurements of peak memory consumption and Time-To-First-Token (TTFT) during the prefill stage, conducted on a single NVIDIA A100-80GB GPU using PyTorch. The experiments averaged over five runs with three warmup iterations, compare the performance of memory-unconstrained (Case 1, 2) and memory-constrained (Case 3) approaches across various context lengths. For memory-unconstrained methods, we observe a linear growth in memory requirements, escalating from 21.29 GB at 5K tokens to 79.38 GB at 100K tokens, accompanied by a proportional increase in TTFT from 0.98 to 3.27 seconds.

Our memory-constrained continual KV cache compression (Case 3) exhibits remarkably different behavior. Despite the increasing context length, the peak memory usage shows only minimal growth, rising modestly from 20.93 GB at 5K tokens to 22.85 GB at 100K tokens. Similarly, the TTFT remains relatively stable, increasing from 1.08 to 1.20 seconds across the same range. These detailed measurements demonstrate that our approach effectively maintains near-constant resource utilization while processing extended video frames.

F Related Work

F.1 MLLMs for Long Video Understanding

Recent advances in long-context MLLMs have attracted significant attention. Notable examples include Gemini-2.0 [34], supporting streaming video; LongVILA [5], capable of handling up to 6,000 video frames; LLaVA-Next-Video [52], which leverages high-quality synthetic instruction data; and Qwen-2-VL [43], enabling hour-long video analysis via multimodal RoPE.

F.2 Input-Vision Compression (IVC)

To address the computational demands of long-form video processing, several approaches have been proposed to compress redundant visual information before it enters the backbone LLM.

LongVU [36] adopts query-dependent input frame sampling and redundant pixel removal for fine-grained video understanding, but the two-tower vision encoding results in high latency during input sampling, making it impractical for streaming scenarios. Additionally, this approach requires training specialized models to operate in the proposed manner, limiting its applicability to existing pre-trained models.

DyCoke [40] reduce redundancies between adjacent frames at the input video level and dynamically updates query-related tokens in the KV cache from external storage. Slow-Fast-LLaVA-1.5 [46] proposes dividing input video processing into separate slow and fast pathways, using different projection methods to reduce input vision tokens. However, this approach still suffers from the limitation of requiring all input vision tokens to be processed simultaneously and necessitates additional model training.

F.3 KV Cache Compression (KVC)

Understanding the long context in MLLMs demands efficient KV cache control to manage memory growth and latency overhead. KV cache compression methods can be broadly categorized into query-dependent and query-agnostic approaches.

Query-Dependent KV cache Compression. Methods like SnapKV [24], H2O [53], HeadKV [12] and ThinK [47] leverage query-to-context attention scores to identify crucial KV entries but require the full context to be prefilled before compression, making them impractical under memory constraints. In the multimodal domain, FastV [4] accelerates prefill by pruning vision tokens at certain layers based on their attention scores from the final query token. SparseVLM [51] selects visual tokens relevant to user queries via cross-attention. Overall, query-dependent methods effectively compress context but struggle to handle diverse queries for the given context after compression [39]. ReKV [35] addresses streaming video scenarios by offloading video-related KV cache to CPU memory and retrieving query-dependent cache entries on demand. This approach relies on external storage and suffers from data transfer overhead, making it unsuitable for memory-constrained streaming video understanding.

Query-Agnostic KV cache Compression. Recent works pursue query-agnostic KV cache compression to eliminate reliance on future queries [13, 8, 16, 20, 6, 32]. In particular, SqueezedAttention [16] uses key-based clustering but requires full-context encoding, limiting its applicability to memory-constrained settings. InfiniPot [21] compresses context by approximating potential user queries through a task-specific proxy prompt, but it's fixed prompt restricts flexibility. In the vision domain, HiRED [1] and FasterVLM [50] utilize [CLS] token attention scores for compression decisions. However, their reliance on special tokens restricts their application to recent MLLMs that lack such tokens [48, 52], limiting their broader applicability.

Case	Streaming MC	Streaming QA	Compression Method	Prefill Budget	Decoding Budget	VideoMME Short	VideoMME Medium	VideoMME Long	VideoMME Avg.	MLVU Holistic	MLVU Single	MLVU Multi	MLVU Avg.	LVB	Avg.
Qwen-2-VL-7B															
	-	-	Full KV	50K	50K	74.68	62.11	55.00	63.93	76.34	73.91	43.29	65.85	58.77	62.85
Case 1	✗	✗	FastV [4] ($L = 2$)	48/3K ($R = 2.8$)		54.11	50.11	48.67	50.96	69.59	59.40	33.84	55.01	47.94	51.30
				48/6K ($R = 5.8$)		59.67	54.55	50.78	55.00	72.00	64.08	33.47	57.60	50.53	54.38
			SnapKV [24]	50K	3K	74.00	61.00	54.22	63.07	77.08	67.49	39.07	62.11	59.06	61.42
				50K	6K	74.22	60.55	54.33	63.03	77.59	73.91	42.90	66.10	58.80	62.64
Case 2	✗	✓	Uniform	50K	3K	70.33	54.67	49.55	58.18	72.29	59.06	33.51	55.54	59.80	57.84
				50K	6K	72.00	58.78	52.11	60.96	77.08	67.49	39.07	62.11	59.11	60.73
			SnapKV†	50K	3K	69.00	54.00	50.67	57.89	75.88	63.48	35.35	58.99	56.70	57.86
				50K	6K	72.11	57.56	52.22	60.63	76.46	66.43	36.22	60.66	56.72	59.34
Case 3 (CKV)	✓	✓	Uniform	3K	3K	66.00	52.44	48.00	55.48	72.54	59.00	33.51	55.59	55.21	55.43
				6K	6K	72.33	53.33	48.67	58.11	72.55	62.19	33.67	57.00	55.82	56.98
				12K	12K	74.00	55.33	51.44	60.26	75.94	65.53	37.01	60.36	57.91	59.51
				24K	24K	74.22	59.22	53.22	62.22	77.22	71.10	40.78	64.18	58.60	61.67
			SnapKV‡	3K	3K	66.67	52.22	49.89	56.26	75.88	63.48	35.35	58.99	54.91	56.72
				6K	6K	72.00	55.33	51.33	59.55	76.46	66.43	36.22	60.66	55.15	58.45
				12K	12K	74.44	58.89	52.89	62.07	75.71	68.61	35.98	61.31	56.89	60.09
				24K	24K	74.22	61.00	53.78	63.00	77.66	71.82	39.90	64.37	59.09	62.15
			InfiniPot [21]	3K	3K	67.11	54.55	51.00	57.55	74.94	61.80	36.60	58.36	54.00	56.64
				6K	6K	72.89	57.33	51.33	60.52	75.02	63.18	37.09	59.11	54.64	58.09
				12K	12K	74.00	57.78	53.22	61.67	74.46	66.46	38.30	60.70	56.94	59.77
				24K	24K	74.22	60.55	53.56	62.78	76.03	71.11	40.29	63.71	57.85	61.44
			InfiniPot-V	3K	3K	73.89	57.78	51.78	61.11	77.73	70.38	43.15	64.70	57.64	61.15
				6K	6K	74.11	60.78	53.44	62.78	77.16	72.31	44.75	65.82	58.40	62.33
				12K	12K	74.22	62.68	53.89	63.59	76.90	73.41	43.97	65.99	59.18	62.92
				24K	24K	74.22	63.22	53.11	63.52	76.91	73.97	42.18	65.73	58.94	62.73
LLaVA-Next-Video-7B															
	-	-	Full KV	25K	25K	74.33	60.11	54.11	62.85	80.60	73.73	49.43	68.75	63.55	65.05
Case 1	✗	✗	Uniform	25K	3K	74.33	62.33	55.00	63.89	80.29	72.38	49.19	68.01	62.35	64.75
				25K	6K	73.89	62.00	54.78	63.56	80.66	72.25	49.62	68.19	62.55	64.76
			SnapKV [24]	25K	3K	74.44	59.89	53.78	62.70	80.41	73.01	49.67	68.46	62.34	64.50
				25K	6K	74.44	60.11	53.78	62.78	80.60	73.45	49.48	68.64	62.34	64.59
Case 2	✗	✓	Uniform	25K	3K	66.33	54.00	49.67	56.67	75.12	59.65	38.55	58.04	59.14	57.95
				25K	6K	71.00	56.33	51.55	59.63	77.84	65.60	43.92	62.90	61.69	61.41
			SnapKV†	25K	3K	64.00	54.55	51.11	56.55	78.53	59.73	41.69	59.94	56.19	57.56
				25K	6K	69.55	58.44	52.78	60.26	80.86	63.65	45.07	63.26	59.90	61.14
Case 3 (CKV)	✓	✓	Uniform	1.5K	1.5K	56.22	46.89	44.00	49.04	69.72	52.53	36.53	52.87	54.92	52.28
				3K	3K	59.22	51.55	47.44	52.74	74.30	57.25	36.48	56.19	54.40	54.44
				6K	6K	64.89	55.67	49.78	56.78	76.71	61.14	34.55	57.99	57.72	57.50
				12K	12K	72.67	59.89	53.00	61.85	80.03	67.33	44.31	64.38	61.04	62.42
			SnapKV‡	1.5K	1.5K	52.40	58.00	51.33	47.89	74.92	56.89	32.62	55.11	53.65	52.22
				3K	3K	62.11	54.55	48.55	55.07	76.94	59.18	35.71	57.55	54.71	55.78
				6K	6K	66.33	56.11	51.11	57.85	79.60	62.15	37.12	59.98	57.81	58.55
				12K	12K	72.11	58.00	53.11	61.07	79.71	67.99	44.89	64.74	58.83	61.55
			InfiniPot [21]	1.5K	1.5K	53.22	51.11	47.55	53.11	69.89	56.44	30.54	52.88	52.14	52.71
				3K	3K	58.22	51.78	49.33	54.22	72.42	55.88	34.45	54.48	52.43	53.71
				6K	6K	62.44	53.89	51.11	55.81	76.46	57.97	37.07	57.28	55.58	56.22
				12K	12K	70.55	59.22	52.55	60.77	79.84	67.81	45.57	64.89	59.23	61.63
			InfiniPot-V	1.5K	1.5K	63.89	52.55	47.11	54.52	77.08	57.32	34.64	56.49	56.48	55.83
				3K	3K	67.78	56.22	50.33	58.11	77.88	65.74	40.31	61.94	58.37	59.47
				6K	6K	72.44	59.55	51.33	61.11	80.03	69.41	43.93	65.16	60.86	62.38
				12K	12K	73.89	58.67	52.11	61.55	80.91	71.16	51.57	68.35	61.84	63.91

Table A5: **InfiniPot-V vs KVC** Offline long video understanding evaluation results under memory-constrained scenario (case 3), with MC (Memory-Constrained) and QA (Query-Agnostic) conditions marked. Results are reported on (1) Video-MME - Short: -3min, Medium: 3-30min, Long: 30min-2h, (2) MLVU - Holistic, Single-Detail, Multi-Detail LVU, and (3) LVB (LongVideoBenchmark).

G Experimental Results Data

G.1 Comparison between InfiniPot-V and KVC

Tab. A5 provides a detailed performance comparison between KV cache compression (KVC) methods and InfiniPot-V across offline video understanding (OVU) benchmarks under various compression ratios for two models: Qwen-2-VL and LLaVA-Next.

In Case 1, where the full prefill is conducted and the final query is accessible at compression time, FastV demonstrates significantly inferior performance at similar compression ratios due to its aggressive token-pruning strategy. In contrast, SnapKV shows robust performance at high compression ratios across both models by utilizing the full context KV cache and retaining vision tokens that are highly correlated with the given query.

Qwen-2-VL	Vision	Decoding	MLVU				VideoMME				
IVC Methods	Budget	Budget	Holistic	Single	Multi	Avg.	Short	Med	Long	Avg.	Avg.
Full KV	50K	50K	76.3	73.9	43.3	65.9	74.7	62.1	55.0	63.9	64.2
Uniform	50K	6K	77.7	69.8	41.6	64.0	74.9	58.0	52.8	61.9	62.5
TTM [40]	50K	6K	78.2	70.0	42.7	64.5	74.9	59.2	52.7	62.3	62.9
STC [36]	50K	6K	77.9	71.5	44.7	65.7	74.3	59.6	54.6	**62.8**	**63.8**
InfiniPot-V	6K	6K	77.2	72.3	44.7	**65.8**	74.1	60.8	53.4	**62.8**	**63.8**
Uniform	50K	3K	75.7	66.5	38.6	61.1	72.2	53.4	50.0	58.6	59.4
TTM [40]	50K	3K	77.3	67.8	39.5	62.4	72.7	56.2	52.2	60.4	61.0
STC [36]	50K	3K	76.9	68.2	41.7	63.1	71.2	55.9	53.7	60.3	61.3
InfiniPot-V	3K	3K	77.7	70.4	43.2	**64.7**	73.9	57.8	51.8	**61.1**	**62.5**

Table A6: **InfiniPot-V vs IVC**: Performance comparison between Input-Vision Compression (IVC) methodology and InfiniPot-V. Vision budget denotes the vision token length before IVC, while decoding budget refers to the input token length used during decoding. Evaluated using Qwen-2-VL with MLVU and VideoMME datasets.

Case 2 examines the query-agnostic setting, where, as explored in our earlier case study in Appendix. D.2, SnapKV exhibits notable performance degradation across both models when applied in a query-agnostic manner, showing performance comparable to uniform selection baseline.

In Case 3, which represents the CKV framework scenario where the constrained memory budget is used for both prefill and decoding stages, InfiniPot-V significantly outperforms all three baselines across various compression ratios on both models, as showcased in Fig. 2.

G.2 Comparison between InfiniPot-V and IVC

Table A6 presents a performance comparison between Input-Vision Compression (IVC) methods and InfiniPot-V on the MLVU and Video-MME benchmarks using the Qwen-2-VL model. Under a 6K decoding budget, the IVC methods demonstrate robust overall performance by utilizing the full vision encoding budget (50K tokens). InfiniPot-V achieves comparable or slightly superior performance to these methods while operating under constrained memory budgets for both vision encoding and decoding stages (6K tokens).

When the decoding budget is compressed to 3K tokens, the IVC methods exhibit performance degradation, with LongVU's STC methodology achieving the highest performance among the IVC approaches. Notably, InfiniPot-V demonstrates both efficiency and effectiveness by achieving higher accuracy than IVC methods that utilize the full vision encoding budget, while operating under constrained budgets (3K) for both vision encoding and decoding stages.

H Limitation and Future Work

InfiniPot-V introduces the first training-free, query-agnostic framework for memory-constrained streaming video understanding, enabling length-independent KV cache compression with minimal accuracy loss across long-form, real-time scenarios. However, several avenues exist for further advancement. Current approaches focus primarily on vision tokens, yet real-world streaming applications involve multiple modalities including speech, text, and video simultaneously. Future work could extend our framework to unified multimodal compression, enabling more realistic and comprehensive streaming understanding systems that efficiently manage diverse input types within fixed memory constraints.

Additionally, our current fixed budget allocation between TaR and VaN components could benefit from adaptive mechanisms that dynamically adjust compression ratios based on input characteristics—allocating more resources to temporal redundancy reduction for static scenes or prioritizing spatial importance for content-rich frames. Furthermore, while InfiniPot-V's training-free nature ensures broad applicability, end-to-end learning approaches could optimize models specifically for continual compression scenarios, potentially enabling more aggressive compression ratios through learned token importance estimation [18] tailored to streaming video understanding tasks.