EVI-PREFILL: EVIDENCE-AWARE PREFILL FOR EFFICIENT LONG VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing acceleration methods for long video large models adopt a *passive prefill* paradigm that uniformly processes video tokens. This ignores the sequential arrival of task-relevant *visual evidence*: once sufficient evidence emerges early, models still *densely* process the full video stream, leading to redundant computation. To extend video temporal coverage, chunk-wise inference adopts partial observability yet remains *dense*. In this paper, we present **Evi-Prefill**, an *evidence-aware active prefill* framework allocating subsequent compute from visual evidence accumulated under partial observability. Specifically, we formulate evidence gathering for incoming video chunks as a Partially Observable Markov Decision Process. With this model, we estimate evidence demand via the model's internal signals and actively switch from *dense to sparse* inference, reducing redundant compute while preserving performance. Moreover, Evi-Prefill filters structurally redundant frames before visual encoding to boost evidence estimation efficiency. Evi-Prefill requires no training and complements prior methods. Evaluations over 3 benchmarks achieve 1.5–1.9× speedup without average performance degradation, opening a new route to efficient long video understanding via active visual-evidence acquisition. Code will be released.

1 INTRODUCTION

Long Video Large Models (LVLMs) (Yang et al., 2026; An et al., 2026) leverage large language models (LLMs) (Yang et al., 2024) for visual perception, temporal integration, and reasoning, yet they suffer from severe computational bottlenecks. Processing massive video frames incurs high visual encoding cost and produces long visual token sequences, which substantially increase the prefill overhead (Jin et al., 2025; Shao et al., 2026) of LLMs. For hour-long videos (Song et al., 2024; Mangalam et al., 2023; Wang et al., 2025), such overhead often exceeds the latency and GPU memory limits feasible for practical deployment.

Existing research tackles this challenge from two main directions. *Memory-based reasoning* (He et al., 2024; Xie et al., 2026) frameworks (*e.g.*, FlexMem (Chen et al., 2026)) process videos chunk-wise and maintain compact visual memory to improve temporal scalability, enabling LVLMs to handle thousands of frames under limited memory. Meanwhile, *visual compression* (Liu et al., 2025a; Shen et al., 2025) methods remove redundant frames (Sun et al., 2025; Wu et al., 2026) before inference or compress visual representations (Fu et al., 2025b; Lin et al., 2026) during inference to reduce computation latency. Despite promising performance, these approaches all follow a **passive prefill** paradigm: computation scheduling is fixed before the model knows how much task-relevant information has been acquired. As illustrated in Fig. 1-a, all incoming video tokens are prefilled uniformly, ignoring the fact that task-relevant visual evidence arrives sequentially over time. Consequently, even when sufficient evidence is obtained at early stages, the model still processes the full video stream *densely* and introduces unnecessary computation.

This motivates us to shift from passive prefill to **active prefill** (Fig. 1-b), where the LVLM dynamically allocates future compute based on visual evidence accumulated during reasoning. Nevertheless, realizing such evidence-aware compute allocation in chunk-wise inference poses two critical challenges. ❶ *Chunk-wise inference introduces a partial observability problem.* When processing the current chunk, the model cannot access future video content and must decide whether further dense processing is needed solely from previously observed segments. Judging evidence sufficiency

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

becomes a sequential decision-making problem under incomplete information. ❷ *Long video reasoning suffers from low evidence density before visual encoding*. As shown in Fig. 2-b, denser sampling improves temporal coverage of novel frames yet inevitably attenuates latent visual evidence density and introduces redundant information (Wang et al., 2026) before encoding.

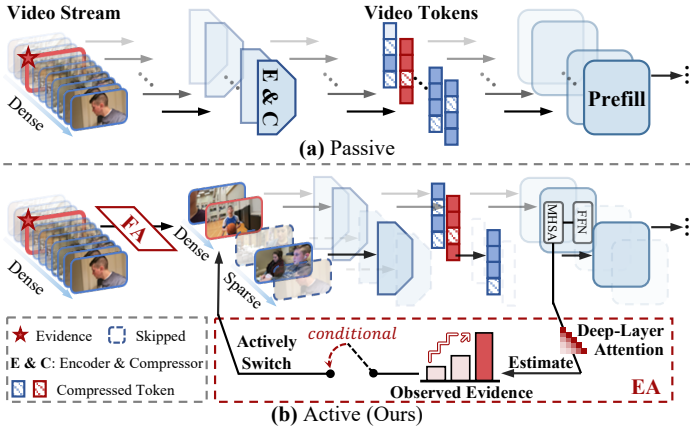

Figure 1: **Comparison between passive and active prefill**. (a) *Passive*: densely process incoming video tokens, overlooking the temporal emergence of visual evidence. (b) *Active (Ours)*: actively switch from dense to sparse processing based on observed evidence. EA and FA denote the proposed Evidence-aware Allocation (§4.1) and Frame Allocation (§4.2), respectively.

To tackle ❶, we analyze the vision-text attention in chunk-wise inference and investigate whether LVLMs provide internal signals for active decision-making (§3.2). We find that deep-layer attention received by newly arrived visual tokens correlates with their marginal evidence contribution (Fig. 2-a). This observation indicates that LVLMs implicitly encode evidence demand in their internal representations, inspiring our Evidence-aware Allocation (EA) strategy (§4.1). By estimating evidence demand, the strategy actively switches inference from dense to sparse and reduces redundant computation. To address ❷, we observe that redundant frames mostly reside in low-attention regions (Fig. 2-c), which overlap with structurally similar visual regions (Appendix §A.1). This suggests that excessive sampling dilutes visual evidence and raises encoding overhead. We thus propose Frame Allocation (FA) (§4.2), which filters structurally redundant frames before encoding to boost evidence density without external models.

These two components form Evi-Prefill, an evidence-aware active prefill framework. It automatically identifies when dense computation is unnecessary and determines which visual observations deserve expensive encoding. Evi-Prefill requires no training and complements existing visual token compression methods. We evaluate Evi-Prefill on 3 long video understanding benchmarks. When combined with mainstream compression approaches, Evi-Prefill speeds up visual encoding and prefill by 1.5–1.9× without sacrificing average performance. Our contributions are three-fold:

- We identify passive prefill as an overlooked issue in prior efficient LVLM methods and introduce the active prefill paradigm, which dynamically allocates future compute based on task-relevant visual evidence accumulated over inference.
- We propose Evi-Prefill, a training-free evidence-aware active prefill framework. It leverages Evidence-aware Allocation to estimate evidence demand under partial observability, and Frame Allocation to raise evidence density.
- Extensive experiments across diverse LVLMs and long video benchmarks show that Evi-Prefill can actively reduce redundant computation while preserving visual evidence coverage, providing a new perspective for efficient long video understanding.

2 Related Work

LVLMs. Video LLMs extend image-based multimodal LLMs (Liu et al., 2023; Chen et al., 2024b) to video understanding through improved visual-language alignment (Ma et al., 2026), temporal modeling, and multimodal instruction tuning. Early models such as Video-LLaVA (Lin et al., 2024) learn unified representations for images and videos, while LLaVA-Interleave (Li et al., 2025b) and LongVA (Zhang et al., 2024) further enhance video understanding through stronger visual representations and cross-scenario transfer. More recent LVLMs (Li et al., 2025a;c; Team et al., 2026) further extend temporal coverage and reasoning capabilities to increasingly long visual contexts. However, scaling LVLMs to long videos introduces substantial visual encoding and LLM computation due to the rapidly growing number of visual tokens, motivating the development of more efficient LVLMs.

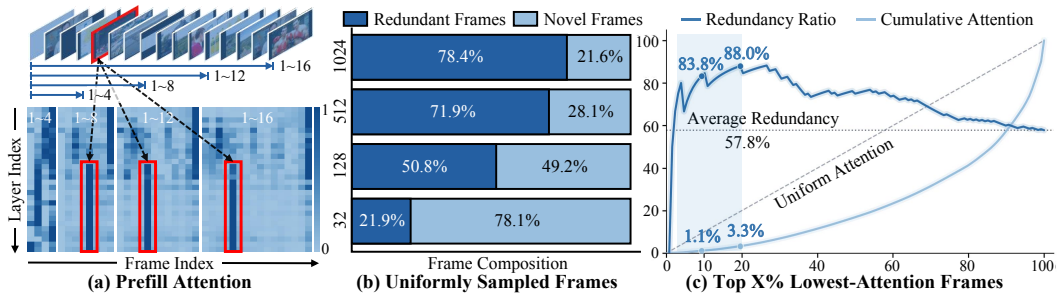

Figure 2: **Visual evidence is unevenly distributed across chunks and frames.** (a) Question-conditioned prefill attention over streaming chunks can reveal evidence demand. (b) Dense sampling incurs increasing frame redundancy with more sampled frames. (c) Such redundancy concentrates in the low-attention tail (blue shaded region): low-mass frames contain substantial redundancy.

Efficient LVLMs. To improve the efficiency of LVLMs, existing methods can be broadly grouped into three directions. *Visual-token compression* methods (Chen et al., 2024a; Tao et al., 2025) prune or merge redundant tokens after visual encoding or within the LLM, reducing downstream computation under a fixed frame budget. *Frame-selection* methods (Tang et al., 2025; Zhu et al., 2026) reduce visual encoding cost by selecting informative or query-relevant frames, often relying on auxiliary models such as CLIP (Radford et al., 2021) or BLIP (Li et al., 2022) and introducing an additional screening stage. Recent *memory-based reasoning* methods (He et al., 2024; Xie et al., 2026; Chen et al., 2026) process videos incrementally and maintain compact historical representations, enabling LVLMs to handle longer temporal contexts under bounded GPU memory.

These approaches primarily optimize *what to retain or compress* within a given video stream, while still following a densely passive prefill paradigm. In contrast, Evi-Prefill focuses on *when continued dense processing is necessary*, leveraging evidence accumulated during prefill to actively adapt computation for future video content.

3 Preliminaries and Observations

3.1 Preliminaries

Memory-Based Passive Chunked Inference. We construct a vanilla memory-based chunked inference baseline (Chen et al., 2026) to expose its inherent passive computational characteristics, which consists of four stages: visual encoding, chunk-wise prefilling, visual-memory writing, and retrieval-based decoding. Let a sampled video $V = \{x_i\}_{i=1}^{N}$ be divided into $B = \lceil N/c \rceil$ consecutive chunks $\mathcal{C}_b$, each containing at most c frames. A vision encoder E and multimodal projector P produce $Z_b = P(E(\mathcal{C}_b))$. Here, Z_b denotes the visual tokens generated from $\mathcal{C}_b$. If each frame produces m visual tokens without compression, then $|Z_b| \approx m|\mathcal{C}_b|$, $\sum_{b=1}^{B} |\mathcal{C}_b| = N$. The language model then incrementally prefills $\mathcal{F}_{\text{pre}}(\cdot)$ the question Q, the current visual tokens, the k retained context memories $(R_{b-1}, ..., R_{b-k})$, and the existing memory M_{b-1}, yielding hidden states H_b used to update the memory:

$$H_b = \mathcal{F}_{\text{pre}}(Q, Z_b, R_{b-1}, ..., R_{b-k}, M_{b-1}), \qquad M_b = \mathcal{U}(M_{b-1}, H_b), \tag{1}$$

where $\mathcal{U}(\cdot)$ denotes the memory-update operation. After all chunks have been read, the baseline retrieves question-relevant memory $\widehat{M} = \mathcal{R}(M_B, Q)$ and decodes the answer $\widehat{y} = \mathcal{D}(Q, \widehat{M})$.

Computation Bottlenecks. Despite reducing peak memory via chunk-wise processing and token compression, this paradigm remains *passive*: every frame is encoded and every compressed token is prefilled, regardless of evidence demand. Its computation cost can be decomposed as

$$\mathcal{T}_{\text{base}} = \sum_{b=1}^{B} \left[\mathcal{T}_{\text{enc}}(|\mathcal{C}_b|) + \mathcal{T}_{\text{pre}}(|Z_b|) + \mathcal{T}_{\text{write}}^{(b)} \right] + \mathcal{T}_{\text{ret}} + \mathcal{T}_{\text{dec}}, \tag{2}$$

where $\mathcal{T}_{\text{enc}}$, $\mathcal{T}_{\text{pre}}$, and $\mathcal{T}_{\text{write}}$ denote the costs of visual encoding, chunk-wise prefilling, and memory writing for each chunk, respectively, while $\mathcal{T}_{\text{ret}}$ and $\mathcal{T}_{\text{dec}}$ represent retrieval and answer decoding

costs. The two costs that scale with video length are visual encoding, with $\mathcal{T}_{\text{enc}}^{\text{total}} \propto \sum_{b=1}^{B} |\mathcal{C}_b| = N$, and chunk-wise prefilling, with $\mathcal{T}_{\text{pre}}^{\text{total}} = \sum_{b=1}^{B} \mathcal{T}_{\text{pre}}(|Z_b|)$. This reveals a mismatch between uniform computation allocation and non-uniform evidence utility, raising a fundamental question: *how should visual evidence perception be guided, and at what rate should evidence be acquired?*

3.2 Key Observations

Based on the above computation bottlenecks, we investigate whether LVLMs under chunk-wise inference inherently provide signals for evidence-aware computation allocation. Specifically, we analyze the visual-text attention from deep Transformer layers, aggregating question-to-visual attention as an observable proxy for how visual information is utilized during inference.

i) New Chunk Attention Reveals Evidence Demand. During chunk-wise inference, we observe that the attention assigned to newly arriving chunks changes according to the remaining evidence required by the query. As shown in Fig. 2-a, for questions requiring localized evidence, attention quickly concentrates on relevant chunks and diminishes after sufficient evidence has been observed. In contrast, questions involving comparison or global temporal reasoning maintain higher attention on later chunks (see Appendix §A.1). This indicates that LVLMs implicitly estimate the marginal utility and demand of incoming visual evidence, suggesting that new-chunk attention can serve as an internal signal for adaptive temporal allocation.

ii) Frame Redundancy Dilutes Evidence Density. Increasing temporal coverage improves evidence capture, but rapidly increases frame redundancy under uniform sampling as shown in Fig. 2-b. We further find that such redundancy is highly concentrated in the low-attention tail (Fig. 2-c): frames accounting for only a small fraction of cumulative attention mass contain a disproportionate amount of redundant content. This reveals a mismatch between sampling density and effective evidence density. The overlap between frame-similarity redundancy and low-attention regions (see Appendix §A.1) further motivates structural redundancy filtering before visual encoding.

4 Method

The above observations (§3.2) reveal that visual evidence is unevenly distributed across both temporal chunks and sampled frames. Thus we introduce **Evi-Prefill**, an evidence-aware active prefill framework that transforms passive chunk-wise inference into adaptive evidence acquisition through EA (§4.1) and FA (§4.2) for evidence density enhancement. Unlike conventional LVLM inference, where computation is uniformly allocated to incoming video tokens, Evi-Prefill actively determines how much visual content should be processed and which observations deserve expensive encoding.

4.1 Evidence-aware Allocation

Partially Observable Evidence Acquisition. During chunk-wise inference, the model observes only a video prefix and must decide how much computation to allocate to the unseen suffix. Although task-relevant evidence may have already been acquired, critical evidence may still appear later, making this decision inherently partially observable. We therefore view evidence acquisition through the lens of a Partially Observable Markov Decision Process (POMDP) (Åström, 1965; Smallwood & Sondik, 1973). At chunk b, the latent state s_b represents whether the accumulated evidence is sufficient for the current query, the observation o_b is derived from the processed prefix, and the action a_b controls the temporal density of subsequent processing. Rather than learning transition and observation models or explicitly solving the POMDP, Evi-Prefill uses model-internal attention as a training-free surrogate for the belief of evidence sufficiency.

New Chunk Mass as Evidence Observation. Motivated by the observation **i)** (§3.2), we extract the question-conditioned attention assigned to newly arrived visual tokens in last $\mathcal{L}_e$ transformer layers. For a processed chunk $\mathcal{C}_b$ with visual tokens Z_b, we aggregate its received attention α_b from query tokens $\mathcal{Q}_e$ and normalize α_b over the observed prefix:

$$\alpha_b = \sum_{\ell \in \mathcal{L}_e} \frac{1}{H} \sum_{h=1}^{H} \sum_{q \in \mathcal{Q}_e} \sum_{v \in Z_b} A_{q \to v}^{\ell,h}, \qquad m_b = \frac{\alpha_b}{\sum_{i=1}^{b} \alpha_i + \epsilon}. \tag{3}$$

We refer to m_b as *New Chunk Mass* (NCM). NCM can be directly obtained from the same prefill pass used for memory writing and serves as an observable proxy for the marginal evidence contribution of the incoming chunk. A high NCM indicates continued demand for new visual information, whereas persistently low NCM suggests diminishing marginal evidence from dense processing.

Surrogate Belief of Evidence Sufficiency. Ideally, temporal allocation would depend on the belief $\beta_b = P(s_b = \text{sufficient} \mid o_{1:b})$ of whether sufficient evidence has already been acquired, where we use $o_b \equiv m_b$ as the observable evidence signal. Instead of explicitly estimating this posterior, Evi-Prefill uses the recent NCM trajectory as a lightweight surrogate: persistent low NCM increases confidence that dense evidence acquisition has begun to saturate (low evidence demand). We first set a minimum dense prefix $B_{\min} = \min\{B, \max(P, \lceil \rho B \rceil)\}$ to establish initial context, where ρ is the minimum ratio. Evi-Prefill then triggers a mode transition when the latest P NCM observations all fall below τ_e:

$$b_s = \min\{b \in \{B_{\min}, \dots, B-1\} \mid \max_{i=b-P+1,\dots,b} m_i < \tau_e\}. \tag{4}$$

Evi-Prefill actively switches from dense evidence acquisition to sparse continuation at b_s.

Uncertainty-aware Sparse Continuation. A high belief of low remaining evidence demand over the observed prefix does not imply that the unseen suffix is irrelevant. Evi-Prefill therefore avoids irreversible early exit and continues to explore future content sparsely. Let $L = B - b_s$ denote the remaining horizon. We adapt the continuation stride to the transition position:

$$S^\star = \min\left\{ \left\lceil \frac{L}{2} \right\rceil, \text{clip}(\text{round}(S_0 \frac{b_s}{L}); S_{\min}, S_{\max}) \right\}, \tag{5}$$

where $S_{\min} = \max(1, \lfloor S_0/2 \rfloor)$ and $S_{\max} = 2S_0$. An earlier transition leaves a longer unseen horizon and therefore uses denser suffix observations, while a later transition permits more aggressive skipping. The resulting processed chunk set is

$$\Omega = \{1, \dots, b_s\} \cup \{b \in \{b_s + 1, \dots, B-1\} : (b - b_s) \bmod S^\star = 0\} \cup \{B\}. \tag{6}$$

The final chunk is always retained to preserve explicit end-of-video coverage. Only chunks in Ω undergo visual encoding, prefilling, and memory writing, while retrieval and decoding remain unchanged. Thus, Evi-Prefill retains dense computation while incoming chunks continue to provides evidence and switch to sparse observation once evidence acquisition begins to saturate.

4.2 Frame Allocation

EA determines where to switch from dense to sparse inference, but each processed chunk may still contain redundant observations. Before visual encoding, all sampled frames are globally accessible, providing an opportunity to improve the latent evidence density of visual inputs.

Structural Novelty Estimation. Inspired by observation **ii)** (§3.2), we use the Structural Similarity Index Measure (SSIM) (Wang et al., 2004) on grayscale thumbnails of frames to identify structural redundancy before visual encoding. For each candidate frame x_i, we compute its similarity to the most recently retained frame $x_{\pi(i)}$, where $\pi(i)$ denotes the index of the latest retained frame preceding i. A frame is retained when its structural similarity falls below a threshold τ_s:

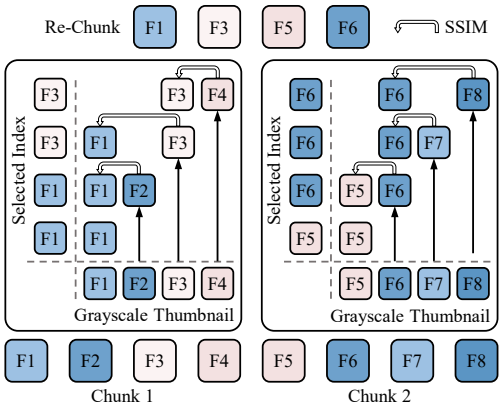

Figure 3: **Illustration of Frame Allocation**. F1–F8 denote the temporally sampled frames.

$$\widehat{V} = \left\{ x_i \in V \;\middle|\; i \in \mathcal{I}_{\text{first}} \text{ or } \text{SSIM}\big(G_r(x_i), G_r(x_{\pi(i)})\big) \le \tau_s \right\}, \tag{7}$$

where $\mathcal{I}_{\text{first}} = \{1 + (b-1)c \mid b = 1, \dots, B\}$ contains the first frame of each original chunk, $\pi(i) = \max\{j < i : x_j \in \widehat{V}\}$ denotes the most recently retained frame, and $G_r(\cdot)$ converts a frame into an $r \times r$ grayscale thumbnail. Thus, highly similar frames are filtered as redundant, while structurally novel observations are preserved, as shown in Fig. 3.

Causal Allocation. During the SSIM threshold pass, the first frame of each original chunk $\mathcal{C}_b$ is retained. Subsequent frames are compared with the latest retained frame rather than their immediate predecessor, allowing gradual visual changes to accumulate into a new retained state instead of being repeatedly discarded. The retained frames are merged chronologically and re-partitioned into $\widehat{B} = \lceil \widehat{N}/c \rceil$ chunks $\{\widehat{\mathcal{C}}_b\}_{b=1}^{\widehat{B}}$, where $\widehat{N} = |\widehat{V}|$ and $\widehat{Z}_b = P(E(\widehat{\mathcal{C}}_b))$. Since selection operates only on low-resolution thumbnails, it requires neither encoding nor an auxiliary model and reduces the number of frames entering the visual encoder from N to $\widehat{N}$.

4.3 Overall Computation Reduction

Evi-Prefill reduces computational cost at both frame and chunk levels. FA reduces (N, B) to $(\widehat{N}, \widehat{B})$ with filtering cost $\mathcal{T}_{\text{filter}}$, after which EA is applied to the filtered stream, yielding $\Omega \subseteq \{1, \ldots, \widehat{B}\}$. Let $N_\Omega = \sum_{b\in\Omega} |\widehat{\mathcal{C}}_b|$ denote the number of frames actually processed. The resulting cost is

$$\mathcal{T}_{\text{Evi}} = \mathcal{T}_{\text{filter}} + \sum_{b\in\Omega} \left[\mathcal{T}_{\text{enc}}(|\widehat{\mathcal{C}}_b|) + \mathcal{T}_{\text{pre}}(|\widehat{Z}_b|) + \mathcal{T}_{\text{write}}^{(b)} \right] + \mathcal{T}_{\text{ret}} + \mathcal{T}_{\text{dec}}. \tag{8}$$

Overall, $(N, B) \to (\widehat{N}, \widehat{B}) \to (N_\Omega, |\Omega|)$, with $N_\Omega \le \widehat{N} \le N$ and $|\Omega| \le \widehat{B} \le B$. Thus, compared with Eq. 2, visual encoding is reduced from N to N_Ω frames, while prefilling and memory writing costs are reduced from B to $|\Omega|$ chunks. See Appendix §A.6 for the algorithm.

5 Experiments

5.1 Experimental Settings

Benchmarks. We evaluate our Evi-Prefill on 3 widely used long video understanding benchmarks: LongVideoBench (Wu et al., 2024), VideoMME (Fu et al., 2025a), and MLVU (Zhou et al., 2025). LongVideoBench and VideoMME are further reported by detailed video-duration subsets, allowing us to examine whether performance remains reliable as the input duration grows. All methods are integrated into LMMS-Eval (Zhang et al., 2025a) and evaluated under the same protocol.

Implementation Details. We evaluate our method on two representative LVLMs: LLaVA-Video (Zhang et al., 2025b) and LLaVA-OneVision (Li et al., 2025a). Unless otherwise specified, all experiments are conducted on NVIDIA RTX 3090 GPUs. All datasets use the same number of sampled frames at a sampling rate of 1 fps. To avoid out-of-memory errors, we employ a carefully designed on-demand frame chunk streaming scheme, which enables our evaluation to scale from 512 frames up to 2048 frames and beyond. Detailed information is provided in Appendix §A.3.

Baselines. We first establish vanilla memory-based passive chunked inference as our baseline (§3.1) which we denote as **ChunkMem**. Representative visual-token compression methods, including VidCom2 (Liu et al., 2025b), VisionZip (Yang et al., 2025), FlashVID (Fan et al., 2026) and FlexMem (Chen et al., 2026), are integrated into this baseline. We combine Evi-Prefill with the above compression methods to validate its effectiveness.

5.2 Main Comparisons

Performance Comparisons. Table. 1 evaluates Evi-Prefill with representative visual-token compression methods across two LVLM backbones, revealing three advantages. (i) **Cross-Backbone Consistency:** Evi-Prefill improves all seven host–backbone combinations by 0.3–0.8 average score without changing their compression ratios. (ii) **Broad Compatibility:** The gains span both pre-LLM and intra-LLM compression, showing that evidence-aware computation allocation complements token-level redundancy reduction. (iii) **Preserved Long Video Capability:** At only 15% visual tokens, FlexMem with Evi-Prefill surpasses dense ChunkMem on both LLaVA-Video (64.7 vs. 64.5) and LLaVA-OneVision (62.7 vs. 61.7), while further improving LongVideoBench by 1.4 overall and 2.8 on the long subset with LLaVA-OneVision.

Efficiency on LLaVA-Video. Table. 2 shows that Evi-Prefill delivers substantial efficiency gains beyond token compression. Across all host methods, it further reduces vision-encoding latency by 34.2–41.0%, generation latency by 33.1–41.7%, and peak GPU memory by 3.5–9.6%, without

Table 1: **Accuracy comparison on long video understanding benchmarks.** Chunk-based methods process at most 512 sampled frames. "Ratio" denotes the retained visual-token ratio. "Med." denotes the medium subset. "+Evi-Prefill" denotes integrating the proposed method. Best results are shown in bold, while blue and red indicate gains and drops over the host compression method.

	Method	Frames	Ratio	VideoMME				LongVideoBench				MLVU	Avg.
				Short	Med.	Long	Overall	Short	Med.	Long	Overall		
LLaVA-Video-7B	Vanilla TMLR'25	32	100%	76.3	59.0	52.2	62.5	68.1	59.2	51.1	58.2	61.3	60.7
	ChunkMem Baseline	512	100%	77.1	61.6	55.2	64.6	70.4	64.1	53.7	61.4	67.4	64.5
	VidCom2 EMNLP'25	512	25%	74.1	61.9	53.8	63.2	65.9	59.0	50.0	57.1	60.3	60.2
	+ Evi-Prefill Ours			**74.7** +0.6	**62.8** +0.9	**53.9** +0.1	**63.8** +0.6	**66.2** +0.3	**60.2** +1.2	**50.2** +0.2	**57.6** +0.5	**60.8** +0.5	**60.7** +0.5
	VisionZip CVPR'25	512	25%	74.4	61.4	53.3	63.1	66.5	58.0	52.1	57.8	60.5	60.5
	+ Evi-Prefill Ours			**75.1** +0.7	**62.4** +1.0	**54.3** +1.0	**64.0** +0.9	**66.5** +0.0	**60.7** +2.7	**52.8** +0.7	**58.9** +1.1	**61.0** +0.5	**61.3** +0.8
	FlashVID ICLR'26	512	25%	76.1	61.0	53.4	63.5	**67.9**	61.9	**53.0**	59.8	62.8	62.0
	+ Evi-Prefill Ours			**76.6** +0.5	**61.4** +0.4	**54.7** +1.3	**64.2** +0.7	67.6 -0.3	**64.1** +2.2	51.8 -1.2	**59.8** +0.0	**63.0** +0.2	**62.3** +0.3
	FlexMem CVPR'26	512	15%	75.6	62.4	55.2	64.4	68.4	**66.0**	55.0	62.0	65.9	64.1
	+ Evi-Prefill Ours			**76.1** +0.5	**63.1** +0.7	**56.3** +1.1	**65.2** +0.8	**68.7** +0.3	65.8 -0.2	**56.0** +1.0	**62.5** +0.5	**66.4** +0.5	**64.7** +0.6
LLaVA-OneVision-7B	Vanilla TMLR'25	32	100%	70.1	56.6	48.7	58.4	70.6	54.4	48.6	56.3	63.1	59.3
	ChunkMem Baseline	512	100%	70.9	59.8	49.9	60.2	69.3	60.7	51.8	59.2	65.8	61.7
	VisionZip CVPR'25	512	25%	70.7	60.0	51.0	60.6	**65.9**	57.0	51.1	56.9	65.1	60.9
	+ Evi-Prefill Ours			**71.1** +0.4	**60.1** +0.1	**52.0** +1.0	**61.0** +0.4	64.2 -1.7	**60.0** +3.0	**52.3** +1.2	**57.9** +1.0	**65.8** +0.7	**61.6** +0.7
	FlashVID ICLR'26	512	25%	**71.2**	58.9	51.3	60.5	66.5	57.5	52.1	57.7	65.3	61.2
	+ Evi-Prefill Ours			70.9 -0.3	**60.2** +1.3	**52.2** +0.9	**61.1** +0.6	**67.6** +1.1	**57.8** +0.3	**52.3** +0.2	**58.1** +0.4	**65.9** +0.6	**61.7** +0.5
	FlexMem CVPR'26	512	15%	**71.4**	**61.4**	50.2	61.0	69.5	59.0	51.1	58.5	66.6	62.0
	+ Evi-Prefill Ours			71.1 -0.3	60.9 -0.5	**51.6** +1.4	**61.2** +0.2	**69.8** +0.3	**59.5** +0.5	**53.9** +2.8	**59.9** +1.4	**67.0** +0.4	**62.7** +0.7

Table 2: **Efficiency comparison on LLaVA-Video-7B**. We report vision-encoding, prefill, memory-writing, and generation latency, GPU peak memory, FLOPs, and average performance for up to 1024 input frames. Blue and red indicate gains and drops over the ChunkMem.

Method	Ratio	Vision Enc. Lat. (s) ↓	Prefill Lat. (s) ↓	Memory Lat. (ms) ↓	Generate Lat. (s) ↓	GPU Peak Mem. (MB) ↓	FLOPs ↓	Perf. ↑
ChunkMem Baseline	100%	*3.8*	*32.1*	*500.4*	*37.0*	*23719.7*	*1553.2*	*64.5*
VidCom2 EMNLP'25	25%	3.8 ↑0.1%	7.6 ↓76.2%	430.8 ↓13.9%	12.3 ↓66.7%	18234.9 ↓23.1%	672.9 ↓56.7%	60.2 -4.3
+ Evi-Prefill Ours		**2.5** ↓33.7%	**5.0** ↓84.4%	**289.4** ↓42.2%	**8.2** ↓77.9%	**17531.8** ↓26.1%	**428.6** ↓72.4%	**60.7** -3.8
VisionZip CVPR'25	25%	3.9 ↑1.8%	7.6 ↓76.4%	433.2 ↓13.4%	12.7 ↓65.8%	19841.2 ↓16.4%	670.9 ↓56.8%	60.5 -4.0
+ Evi-Prefill Ours		**2.3** ↓40.2%	**4.4** ↓86.4%	**250.8** ↓49.9%	**7.4** ↓80.0%	**19053.5** ↓19.7%	**376.9** ↓75.7%	**61.3** -3.2
FlashVID ICLR'26	25%	3.8 ↑0.3%	7.6 ↓76.2%	426.4 ↓14.8%	13.9 ↓62.5%	19851.2 ↓16.3%	677.1 ↓56.4%	62.0 -2.5
+ Evi-Prefill Ours		**2.5** ↓33.3%	**5.0** ↓84.3%	**295.2** ↓41.0%	**9.3** ↓74.9%	**19151.1** ↓19.3%	**434.6** ↓72.0%	**62.3** -2.2
FlexMem CVPR'26	15%	3.9 ↑1.7%	17.4 ↓45.7%	763.4 ↑52.6%	22.4 ↓39.6%	19855.2 ↓16.3%	1372.2 ↓11.7%	64.1 -0.4
+ Evi-Prefill Ours		**2.4** ↓37.9%	**10.7** ↓66.8%	**265.0** ↓47.0%	**13.6** ↓63.3%	**17959.2** ↓24.3%	**856.1** ↓44.9%	**64.7** +0.2

changing token ratios. For 25%-token compressors, FLOPs reduction over dense ChunkMem increases from about 56% to 72.0–75.7%. Even with FlexMem at 15% tokens, Evi-Prefill reduces FLOPs from 1372.2G to 856.1G while improving performance from 64.1 to 64.7. These results show that Evi-Prefill provides an efficiency dimension complementary to token compression by reducing how much visual content needs to be fully processed in the first place.

Efficiency on LLaVA-OneVision. Relative to ChunkMem, Evi-Prefill with FlashVID achieves 1.9×, 7.6×, and 4.9× speedups in vision encoding, prefill and memory writing, and generation, respectively, while reducing peak memory from 23.6 to 19.0 GB. With FlexMem, it achieves 1.5×, 2.8×, and 2.5× speedups, with peak memory reduced to 18.0 GB. These consistent gains across backbones indicate that Evi-Prefill addresses a shared computational bottleneck in long-video inference rather than a backbone-specific inefficiency.

Scaling to Thousands of Frames. Table. 3 scales Evi-Prefill from 512 to 2048 frames on LLaVA-Video-7B. Across all scales, it reduces generation latency by 7.9–8.7 s while matching or outperforming FlexMem on all benchmarks. Despite a 4× increase in input length, latency rises only from 13.2 to 14.5 s, demonstrating strong scalability. Meanwhile, more frames do not yield monotonic gains for FlexMem. This saturation is consistent with prior findings on diminishing returns and attention dilution under excessive visual context (Li et al., 2026).

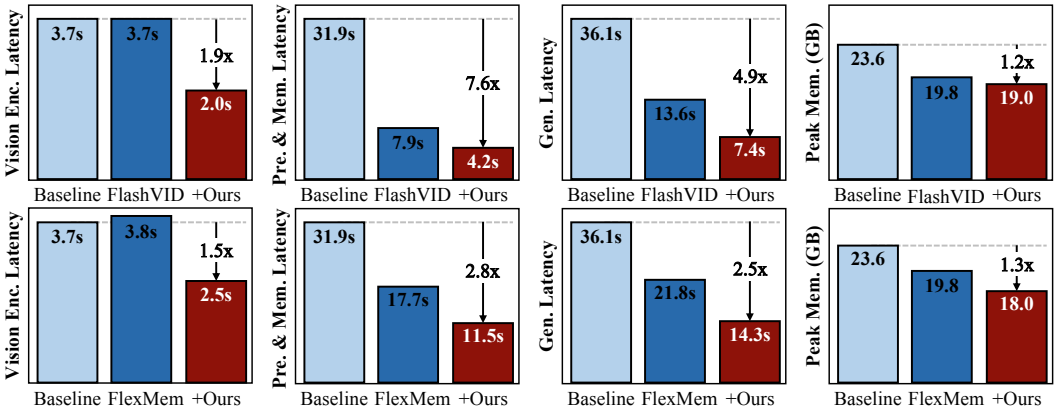

Figure 4: **Efficiency Comparison on LLaVA-OneVision-7B.** We compare FlashVID and FlexMem with and without Evi-Prefill across latency and peak memory usage.

Table 3: **Scaling Performance with Longer Video Inputs**. Results are reported from 512 to 2048 input frames. Generate Lat. = model generation latency.

Method	Frames	MLVU	VideoMME	LongVideo Bench	Generate Lat. (s) ↓
LLaVA-Video-7B					
FlexMem CVPR'26		65.9	64.4	62.0	21.9
+FA	512	66.4	65.3	62.5	15.6
+FA & EA		66.4 +0.5	65.2 +0.8	62.5 +0.5	13.2 ↓8.7
FlexMem CVPR'26		63.5	64.5	60.7	22.4
+FA	1024	63.6	64.3	60.9	17.4
+FA & EA		63.8 +0.3	64.5 +0.0	60.9 +0.2	13.8 ↓8.6
FlexMem CVPR'26		63.8	64.2	61.3	22.4
+FA	2048	63.9	64.5	61.4	18.8
+FA & EA		64.0 +0.2	64.6 +0.4	61.4 +0.1	14.5 ↓7.9

Table 4: **Ablation Study of LLM Layers for NCM Computation on LLaVA-Video-7B**. P & M Lat. = prefill and memory latency.

Layer	VideoMME				LongVideo Bench	P & M Lat. (s) ↓
	Short	Med.	Long	Overall		
VisionZip						
last 1	75.1	62.1	53.7	63.6	**58.9**	**4.1**
last 3	75.1	62.0	54.2	63.8	58.5	**4.1**
last 5	75.1	62.2	54.2	63.9	58.8	4.2
last 7	75.1	**62.4**	**54.3**	**64.0**	**58.9**	4.2
FlexMem						
last 1	**76.7**	62.7	56.0	65.1	61.1	**9.6**
last 3	76.1	62.8	56.0	65.0	**61.5**	10.0
last 5	**76.7**	**63.6**	55.6	**65.3**	61.3	10.2
last 7	76.1	63.1	**56.3**	65.2	61.4	10.2

5.3 Ablation Study and Analysis

Complementary Effects of FA and EA. Table. 3 ablates FA and EA across 512–2048 frames. FA removes structurally redundant frames before encoding, reducing latency while preserving or improving accuracy through denser visual evidence. EA further cuts latency via selective chunk-wise computation while retaining FA's gains; notably, its additional saving grows from 2.4 s at 512 frames to 4.3 s at 2048 frames, indicating increasing chunk-level redundancy with longer contexts. Together, FA and EA act at complementary stages—frame selection and chunk-level computation allocation—and yield stronger joint savings as input length grows.

Evidence-aware Allocation. We ablate three key design choices of EA. *Deep-layer selection.* Table. 4 shows that using the last seven LLM layers yields robust performance across VisionZip and FlexMem with marginal latency overhead, and is adopted by default. *NCM threshold.* As shown in Table. 5, lower thresholds delay the dense-to-sparse transition, revealing an accuracy–efficiency trade-off. Performance saturates at 0.005: lowering it to 0.004 increases processed chunks from 108.1 to 123.5 and latency from 24.7 s to 27.9 s without accuracy gains. *Sparse continuation.* Adaptive stride achieves the best accuracy (56.3 Long / 65.2 Overall) at nearly the cost of hard exit, preserving late evidence without the overhead of fixed-stride sampling.

Table 5: **Ablation study of NCM threshold and sparse continuation**. "Processed Chunks" denotes the number of chunks processed during inference.

Configuration	VideoMME		Processed Chunks ↓	Generate Lat. (s) ↓
	Long	Overall		
New Chunk Mass threshold				
0.008	54.7	64.1	**89.7**	**21.2**
0.006	55.5	64.8	100.1	23.5
0.005	**56.0**	**65.0**	108.1	24.7
0.004	**56.0**	**65.0**	123.5	27.9
Sparse continuation strategy				
Hard exit	56.0	65.0	**108.1**	**24.7**
Fixed stride	55.9	65.0	112.7	25.8
Adaptive stride	**56.3**	**65.2**	110.8	24.9

Frame Allocation. Table. 6 ablates the downsampled size and similarity threshold. *Downsampled size.* Using 224×224 thumbnails cuts selection latency from 11.7 s to 1.8 s (84.6%) with negligible

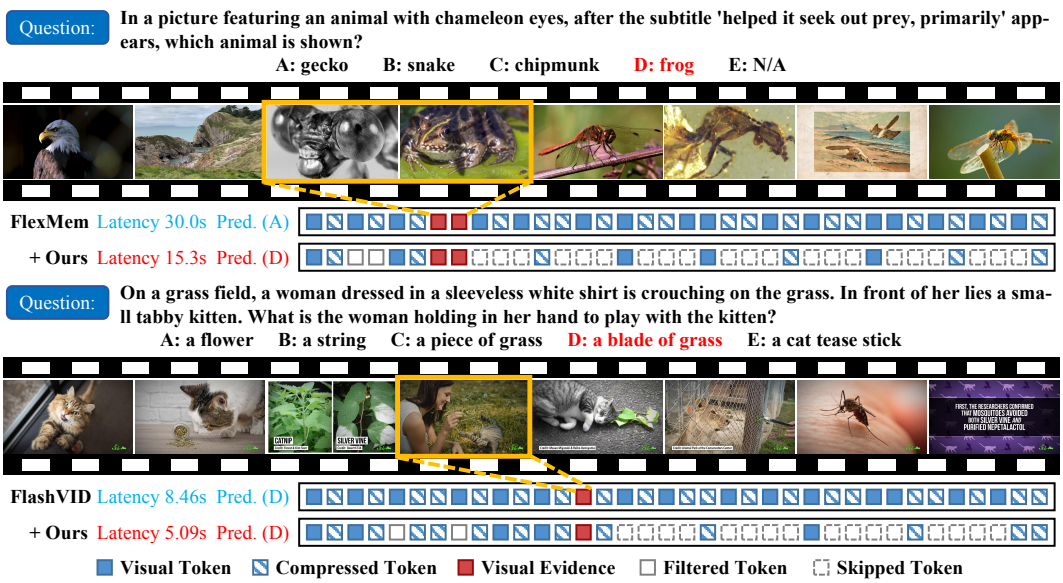

Figure 5: **Qualitative examples of Evi-Prefill integrated with FlexMem and FlashVID.** Evi-Prefill preserves relevant evidence, enables an active dense-to-sparse transition, processes fewer chunks and reduces latency. See more examples in Appendix §A.4.

accuracy loss, showing that coarse structural cues suffice for redundancy detection. Further reducing to 128×128 lowers latency but begins to degrade accuracy. *Similarity threshold.* Lowering τ_s reduces generation latency from 16.0 s to 12.4 s, but overly aggressive filtering sacrifices accuracy. We adopt 224×224 and τ_s=0.85 as a robust trade-off between accuracy and efficiency .

Qualitative Analysis. Fig. 5 illustrates how Evi-Prefill complements token compression through evidence-aware computation. In the FlexMem case, frog evidence appears early; while FlexMem processes the full sequence and predicts incorrectly (A), Evi-Prefill retains the evidence and switches to sparse processing, reducing latency from 30.0 s to 15.3 s while recovering the correct answer (D). In the FlashVID case, Evi-Prefill preserves the key evidence for "a blade of grass" while skipping low-value later chunks, maintaining the correct prediction with latency reduced from 8.46 s to 5.09 s. Together, these cases show that Evi-Prefill operates orthogonally to token compression: the host method compresses retained visual tokens, while Evi-Prefill avoids processing unnecessary visual context.

Table 6: **Ablation study of downsampled size and similarity threshold**. Select Lat. = frame selection overhead.

Configuration	VideoMME	LongVideo Bench	Select Lat. (s) ↓	Generate Lat. (s) ↓
Downsampled size				
Original	**65.3**	**62.5**	11.7	16.2
224	65.2	**62.5**	1.8	14.5
128	64.8	62.4	**1.4**	**12.2**
Similarity threshold				
0.90	64.8	61.7	1.9	16.0
0.85	65.2	**62.5**	**1.8**	14.5
0.80	**65.3**	62.2	**1.8**	13.5
0.75	64.9	61.9	**1.8**	**12.4**

6 Conclusion

We present Evi-Prefill, an evidence-aware active prefill framework for efficient long video understanding that addresses passive prefill's dense processing of all incoming compressed tokens despite evolving visual evidence. Evi-Prefill models chunk-wise visual evidence acquisition as a POMDP, uses model-internal signals to assess evidence demand, and switches from dense to sparse inference. It further filters structurally redundant frames to improve evidence estimation. Across three benchmarks, Evi-Prefill further accelerates existing methods by 1.5–1.9×, while even improving average performance. Our study reveals that visual evidence in long videos is unevenly distributed over time and that active evidence acquisition provides an effective paradigm for adaptive computation allocation in large video models. Future work will explore more fine-grained evidence modeling and adaptive inference strategies for scalable video understanding.

AI USE STATEMENT

We used generative AI tools to assist with the interpretation and presentation of experimental results and to improve the clarity, concision, and organization of the manuscript. Generative AI was not used to generate experimental results or data. All AI-assisted analyses and text were manually reviewed and verified against the experimental results and relevant literature by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of Evi-Prefill and its two components in Method (§4). Algorithms, implementation details, hyperparameters, and additional experimental settings are provided in the Appendix. After our submission is accepted, we will release the complete source code to the public and make it available in an open repository for the community to access, review, and reuse.

REFERENCES

Xiang An, Yin Xie, Feilong Tang, Yunyao Yan, Huajie Tan, Didi Zhu, Changrui Chen, Xiuwei Zhao, Bin Qin, Kaicheng Yang, et al. Llava-onevision-2: Towards next-generation perceptual intelligence. *arXiv preprint arXiv:2605.25979*, 2026.

Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

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. In *ECCV*, 2024a.

Tao Chen, Kun Zhang, Qiong Wu, Xiao Chen, Chao Chang, Xiaoshuai Sun, Yiyi Zhou, and Rongrong Ji. Scaling the long video understanding of multimodal large language models via visual memory mechanism. In *CVPR*, 2026.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024b.

Ziyang Fan, Keyu Chen, Ruilong Xing, Yulin Li, Li Jiang, and Zhuotao Tian. FlashVID: Efficient video large language models via training-free tree-based spatiotemporal token merging. In *ICLR*, 2026.

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

Tianyu Fu, Tengxuan Liu, Qinghao Han, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. Framefusion: Combining similarity and importance for video token reduction on large vision language models. In *ICCV*, 2025b.

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

Yizhang Jin, Jian Li, Tianjun Gu, Yexin Liu, Bo Zhao, Jinxiang Lai, Zhenye Gan, Yabiao Wang, Chengjie Wang, Xin Tan, et al. Efficient multimodal large language models: A survey. *VI*, 3(1): 27, 2025.

Aiden Yiliu Li, Nels Numan, and Anthony Steed. Visual agentic memory: Enabling online long video understanding via online indexing, hierarchical memory, and agentic retrieval. *arXiv preprint arXiv:2605.16481*, 2026.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *TMLR*, 2025a.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-interleave: Tackling multi-image, video, and 3d in large multimodal models. In *ICLR*, 2025b.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *SCIS*, 68(10):200102, 2025c.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In *EMNLP*, 2024.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

Shuming Liu, Chen Zhao, Tianqi Xu, and Bernard Ghanem. Bolt: Boost large vision-language model without training for long-form video understanding. In *CVPR*, 2025a.

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

Qi Ma, Chen-Yang Wang, Dehong Gao, and Deng-Ping Fan. Alignednorm: Prompting vision–language models via coupled prompt field. In *ICML*, 2026.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *NeurIPS*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Kele Shao, Keda TAO, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. A survey of token compression for efficient multimodal large language models. *TMLR*, 2026.

Leqi Shen, Guoqiang Gong, Tao He, Yifeng Zhang, Pengzhang Liu, Sicheng Zhao, et al. Fastvid: Dynamic density pruning for fast video large language models. In *NeurIPS*, 2025.

Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *CVPR*, 2024.

Hui Sun, Shiyin Lu, Huanyu Wang, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Ming Li. Mdp 3: a training-free approach for list-wise frame selection in video-llms. In *ICCV*, 2025.

Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding. In *CVPR*, 2025.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *CVPR*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, et al. Lvbench: An extreme long video understanding benchmark. In *ICCV*, 2025.

Yiyu Wang, Xuyang Liu, Xiyan Gui, Xinying Lin, Boxue Yang, Chenfei Liao, Tailai Chen, and Linfeng Zhang. Accelerating streaming video large language models via hierarchical token compression. In *CVPR*, 2026.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. In *NeurIPS*, 2024.

Zhirong Wu, Xiaodong Wang, Langling Huang, Teng Xu, and Peixi Peng. A training-free framework for long video understanding via video-query-options similarity. In *ICLR*, 2026.

Yiweng Xie, Bo He, Junke Wang, Xiangyu Zheng, Ziyi Ye, and Zuxuan Wu. Fluxmem: Adaptive hierarchical memory for streaming video understanding. In *CVPR*, 2026.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

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

Zuhao Yang, Sudong Wang, Kaichen Zhang, Keming Wu, Sicong Leng, Yifan Zhang, Bo Li, Chengwei Qin, Shijian Lu, Xingxuan Li, and Lidong Bing. Longvt: Incentivizing "thinking with long videos" via native tool calling. In *CVPR*, 2026.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *ACL Findings*, 2025a.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun MA, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *TMLR*, 2025b.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, et al. Mlvu: Benchmarking multi-task long video understanding. In *CVPR*, 2025.

Zirui Zhu, Hailun Xu, Yang Luo, LIU YONG, Kanchan Sarkar, Zhenheng Yang, and Yang You. Focus: Efficient keyframe selection for long video understanding. In *ICLR*, 2026.

A Appendix

A.1 More Observations

We provide additional visual evidence for the two observations in Sec. 3.2. First, the query-dependent use of incoming visual chunks. Second, the overlap between structural frame redundancy and low-attention regions.

New Chunk Attention Reveals Evidence Demand. Fig. 6 compares two questions about the *same* video, thereby holding the visual input constant while changing the evidence required to answer. The localized-evidence question elicits attention concentrated on fewer temporal positions, whereas the global-evidence question sustains responses over a broader set of positions and layers. The contrast illustrates why a fixed processing budget may be unsuitable even for one video: the relevant temporal coverage also depends on the question. The heatmaps visualize frame-level attention of chunk-wise input.

Fig. 7 presents a layer-wise visualization of attention patterns for a 64-frame sample. Attention in many earlier layers is relatively diffuse or position-sensitive, whereas later layers become more selective over frame positions. This motivates extracting NCM from the last few language-model layers, as specified in Sec. 4.1, instead of treating all layers as equally informative. In actual chunk-stream inference, NCM is computed only for a chunk *after* that chunk has been prefilled; sustained low NCM over processed chunks triggers sparse continuation rather than a claim that unseen chunks are irrelevant.

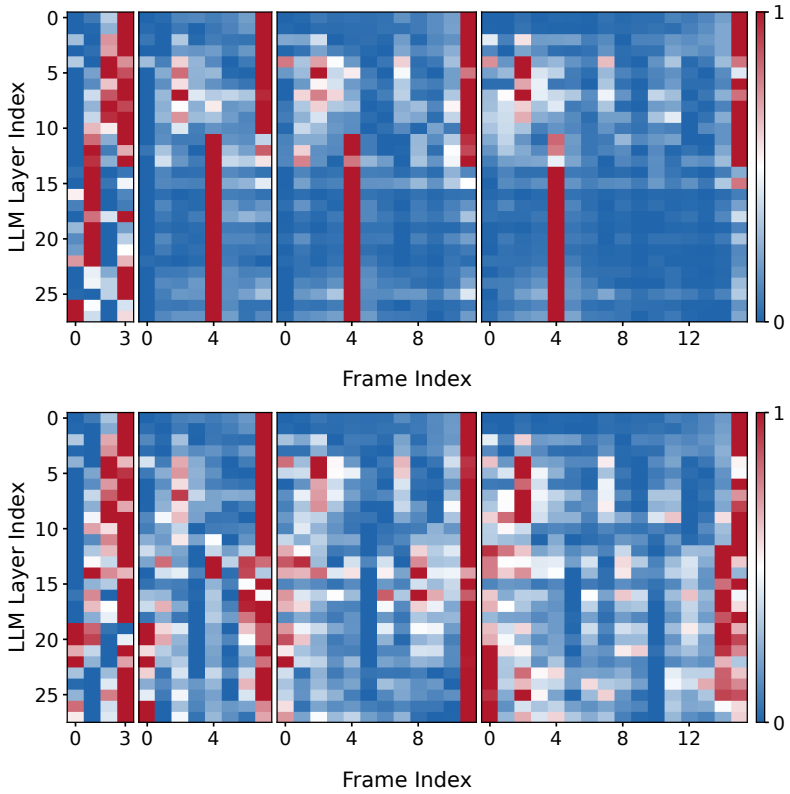

Figure 6: **The same video, two questions with different evidence scopes.** Question-to-frame attention is shown by LLM layer (vertical axis) and sampled frame index (horizontal axis, grouped into successive chunks); warmer colors indicate higher attention. The upper and lower rows correspond to localized- and global-evidence questions, respectively. Holding the video fixed makes the difference in attention allocation attributable to the question, without implying that attention alone certifies answer sufficiency.

Frame Redundancy Dilutes Evidence Density. Fig. 8 aligns the structural-similarity trace with the attention assigned to the same sampled frames. Among the 74 near-duplicates identified by the similarity threshold in this example, 62 (83.8%) receive below-mean frame attention, while 12 receive above-mean attention. The substantial overlap supports the use of inexpensive similarity filtering *before* visual encoding: repeatedly encoding near-identical views consumes resources even when the model allocates little attention to many of them. The 12 exceptions also matter. Structural similarity serves as a fast metric for redundancy measurement before the visual encoder. Many other encoding-agnostic metrics, such as pixel-wise cosine similarity and MSE are also available for measurement.



Figure 7: **Question-to-frame attention across LLM layers.** Each panel shows the normalized attention over 64 sampled frames for one of 28 layers (0–27). Deeper layers exhibit sharper concentration on a subset of frames than many early layers, motivating our use of late-layer attention as the observation for Evidence-aware Allocation.

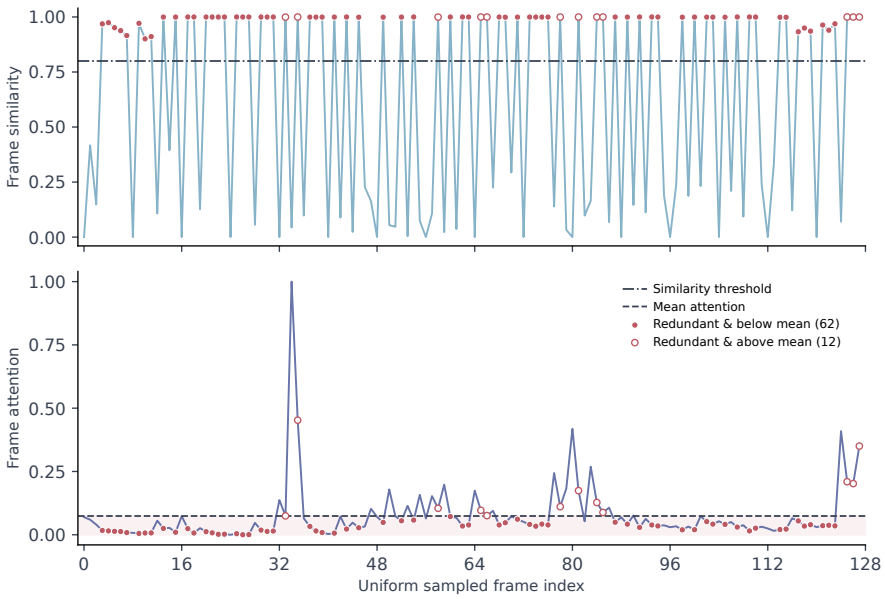

Figure 8: **Structural redundancy and model attention in one video.** Top: frame similarity and the threshold used to mark near-duplicates. Bottom: frame attention and its mean across sampled frames. Of the 74 marked redundant frames, 62 fall below mean attention (filled markers) and 12 remain above it (open markers). Structural similarity thus overlaps substantially with low model attention in this example.

These two observations motivate complementary allocations: Frame Allocation reduces structural repetition within chunks, while Evidence-aware Allocation adapts how densely the resulting chunks are processed according to the model's observed attention to newly arrived content. Sparse continuation preserves periodic and final-chunk coverage because attention from the observed prefix cannot rule out later evidence.

A.2 POMDP Interpretation of Evidence-aware Allocation

Decision process and information boundary. The POMDP view in Sec. 4.1 describes *computation allocation over a fixed video*, not an agent that changes the video. After Frame Allocation, the ordered chunks $\mathcal{C}_{1:B}$ and their number B are fixed, but a future chunk has not been processed by the vision encoder or LLM prefill when a decision about it is made. A conceptual full state at step b would include the query-relevant evidence already represented in memory, the evidence still present in unseen chunks, and whether the accumulated evidence suffices to answer Q. The binary sufficiency variables s_b used in the main text is a decision-relevant summary of this richer state. Processing a chunk advances the prefix and updates visual memory; skipping it advances the video position without encoding, prefilling, or writing that chunk. The video itself remains unchanged under either action.

Observation, action, and objective. The available history contains Q, the processed chunk indices, the current memory, and the attention statistics produced by processed chunks. For a processed chunk, the observation $o_b = m_b$ is its query-conditioned New Chunk Mass from Eq. 3, obtained from the same prefill pass that updates memory. The decision made after processing chunk b applies only to subsequent chunks. No NCM observation is available for a skipped chunk: apart from the earlier CPU thumbnail-based Frame Allocation, EA does not encode or prefill that chunk and cannot use its attention to make an earlier decision. The action is to continue dense processing or, after a switch, process only the chunks selected by the sparse-continuation schedule. Conceptually, the desired policy trades answer quality against the cost of encoding, prefilling, and memory writing

over its processed set Ω. This is a modeling objective, not an objective that we optimize by learning a transition kernel, an observation likelihood, or a reward function.

Surrogate policy and residual uncertainty. Our implementation replaces a calibrated belief $P(s_b = \text{sufficient} \mid o_{1:b})$ with a conservative rule on observed NCM values: it first processes at least $B_{\min}$ chunks densely and switches only when the latest P processed chunks all satisfy $m_i < \tau_e$ (Eq. 4). A single low-attention chunk is therefore insufficient to trigger the switch. This matches the patterns in Appendix A.1: persistently low new-chunk attention after localized evidence can trigger sparse continuation, whereas continuing attention to distributed evidence keeps dense processing active. If the criterion never holds, every filtered chunk is processed. Low NCM indicates diminishing *observed* use of new visual content; it does not prove that no relevant event remains unseen. Accordingly, the switch reduces observation density instead of terminating video reading. The stride in Eq. 5 depends on the switch position b_s and remaining horizon L: an early switch leaves a longer uncertain suffix and induces denser periodic coverage, while the coverage cap and forced final chunk prevent the schedule from relying solely on the observed prefix. Once chosen, the stride is not updated from later NCM values and the implementation does not switch back to dense mode. Thus the method limits the risk of missing evidence between sampled tail chunks. Retrieval and answer decoding use the memory of the processed chunks exactly as in the host model.

Role of Frame Allocation. Frame Allocation acts before this chunk-level decision process. Its query-agnostic SSIM test removes local visual redundancy and determines the filtered horizon B, shrinking the candidate stream for later computation. It is not a measurement of evidence sufficiency and does not supply the EA belief surrogate. This separation explains the two complementary allocations in Evi-Prefill: frame selection reduces redundant visual work, whereas the POMDP-inspired chunk policy controls when dense evidence acquisition should give way to sparse coverage.

A.3 Detailed Experimental Settings

Datasets and metrics. We use the video-only setting of three multiple-choice long-video benchmarks. **Video-MME** (Fu et al., 2025a) contains 2,700 questions over 900 videos spanning short, medium, and long durations; we evaluate its `videomme` test split without audio or subtitle input. **LongVideoBench** (Wu et al., 2024) tests long-range, referred reasoning over videos up to one hour; we use its video-only validation task (`longvideobench_val_v`), rather than the interleaved-subtitle variant. **MLVU** (Zhou et al., 2025) covers diverse video genres and reasoning tasks; we report multiple-choice accuracy on `mlvu_dev`. For Video-MME, the short/medium/long groups follow the dataset annotations. For LongVideoBench, the evaluation adapter groups its 15- and 60-second categories as short, its 600-second category as medium, and its 3,600-second category as long. We use the corresponding LMMS-Eval (Zhang et al., 2025a) answer parsers and report exact-match choice accuracy; all paired methods receive the same task prompts and question options.

Host methods and integration. ChunkMem is our uncompressed chunk-wise visual-memory baseline: it encodes and prefills every sampled frame and uses the same memory retrieval mechanism (Chen et al., 2026). **VidCom2** (Liu et al., 2025b), **VisionZip** (Yang et al., 2025), and **FlashVID** (Fan et al., 2026) are inserted after visual encoding and before language-model prefilling. Our VidCom2 adapter applies its video-token selection to projected visual features; the VisionZip adapter uses vision-encoder attention to retain dominant tokens and merges the remaining contextual tokens; the FlashVID adapter uses attention/diversity selection and spatiotemporal token merging on the vision side. We do not enable FlashVID's optional intra-LLM FastV (Chen et al., 2024a) pruning, so its comparison isolates the vision-side compressor. **FlexMem** (Chen et al., 2026) instead uses its native intra-LLM visual-memory/KV compression and retrieval, without an additional external token compressor. For each host, we compare its chunk-wise implementation with and without Evi-Prefill while preserving the host's visual-token compression operator. Evi-Prefill adds Frame Allocation before visual encoding and Evidence-aware Allocation during chunk prefilling. The retained visual-token ratios reported in Table 1 are 25% for VidCom2, VisionZip, and FlashVID and 15% for FlexMem.

Implementation details. We evaluate LLaVA-Video-7B (Zhang et al., 2025b) and LLaVA-OneVision-7B (Li et al., 2025a) in LMMS-Eval with batch size one and greedy decoding (temperature zero, one beam, at most 1,024 generated tokens). Unless specified otherwise, videos are decoded with Decord, sampled at 1 fps, capped at 512 frames for the main comparison, and par-

Table 7: **Component-wise accuracy breakdown of the main comparison.** Chunk-based methods use at most 512 sampled frames; "Ratio" is the host's visual-token retention ratio.

Method	Frames	Ratio	VideoMME				LongVideoBench				MLVU	Average
			Short	Medium	Long	Overall	Short	Medium	Long	Overall		
LLaVA-Video-7B												
Vanilla	*32*	*100%*	*76.3*	*59.0*	*52.2*	*62.5*	*68.1*	*59.2*	*51.1*	*58.2*	*61.3*	*60.7*
ChunkMem Baseline	512	100%	77.1	61.6	55.2	64.6	70.4	64.1	53.7	61.4	67.4	64.5
VidCom2			74.1	61.9	53.8	63.2	65.9	59.0	50.0	57.1	60.3	60.2
+ FA	512	25%	74.7	62.7	53.6	63.6	66.2	60.0	50.0	57.4	60.8	60.6
+ FA & EA			**74.7**	**62.8**	**53.9**	**63.8**	**66.2**	**60.2**	**50.2**	**57.6**	**60.8**	**60.7**
VisionZip			74.4	61.4	53.3	63.1	66.5	58.0	52.1	57.8	60.5	60.5
+ FA	512	25%	75.1	62.4	54.2	63.9	66.5	59.7	52.5	58.5	**61.2**	61.2
+ FA & EA			**75.1**	**62.4**	**54.3**	**64.0**	**66.5**	**60.7**	**52.8**	**58.9**	61.0	**61.3**
FlashVID			76.1	61.0	53.4	63.5	67.9	61.9	**53.0**	59.8	62.8	62.0
+ FA	512	25%	76.4	**62.2**	54.0	64.2	**67.9**	**64.6**	51.6	**60.0**	**63.1**	**62.4**
+ FA & EA			**76.6**	61.4	**54.7**	**64.2**	67.6	64.1	51.8	59.8	63.0	62.3
FlexMem			75.6	62.4	55.2	64.4	68.4	66.0	55.0	62.0	65.9	64.1
+ FA	512	15%	**76.7**	**63.6**	55.6	**65.3**	68.4	**66.7**	55.7	62.5	66.4	64.7
+ FA & EA			76.1	63.1	**56.3**	65.2	**68.7**	65.8	**56.0**	**62.5**	**66.4**	**64.7**
LLaVA-OneVision-7B												
Vanilla	*32*	*100%*	*70.1*	*56.6*	*48.7*	*58.4*	*70.6*	*54.4*	*48.6*	*56.3*	*63.1*	*59.3*
ChunkMem Baseline	512	100%	70.9	59.8	49.9	60.2	69.3	60.7	51.8	59.2	65.8	61.7
VisionZip			70.7	60.0	51.0	60.6	**65.9**	57.0	51.1	56.9	65.1	60.9
+ FA	512	25%	70.9	**60.8**	50.7	60.8	64.0	60.0	52.3	57.8	65.5	61.4
+ FA & EA			**71.1**	60.1	**52.0**	**61.0**	64.2	**60.0**	**52.3**	**57.9**	**65.8**	**61.6**
FlashVID			**71.2**	58.9	51.3	60.5	66.5	57.5	52.1	57.7	65.3	61.2
+ FA	512	25%	70.8	60.2	**52.3**	61.1	67.6	57.8	52.1	58.0	65.7	61.6
+ FA & EA			70.9	**60.2**	52.2	**61.1**	**67.6**	**57.8**	**52.3**	**58.1**	**65.9**	**61.7**
FlexMem			**71.4**	61.4	50.2	61.0	69.5	59.0	51.1	58.5	66.6	62.0
+ FA	512	15%	71.2	**61.4**	51.1	**61.3**	69.8	59.5	53.0	59.5	67.0	62.6
+ FA & EA			71.1	60.9	**51.6**	61.2	**69.8**	**59.5**	**53.9**	**59.9**	**67.0**	**62.7**

titioned into four-frame chunks; longer-input experiments increase only the frame cap. We use half-precision model weights, average spatial pooling with stride two, and the same prompt template across paired runs. Chunk-memory settings are `PREBLK`$=12$ and `TOPB`$=16$; the native FlexMem configuration additionally uses `PRERATIO`$=4$, `DECRATIO`$=2$, and 182 visual-token slots per frame. Experiments run on NVIDIA RTX 3090 GPUs while the analysis was performed on a single A800 GPU.

For Frame Allocation, SSIM is computed on 224×224 grayscale thumbnails against the most recently retained frame within each original chunk; no fixed frame-retention target is imposed. The resulting frame count is aligned to the four-frame chunk size as described in Sec. 4.2. For Evidence-aware Allocation, the exit query contains the question and answer options, and new-chunk attention is accumulated from the last seven language-model layers. We use a minimum dense-prefix ratio $\rho = 0.2$, patience $P = 2$, and an adaptive sparse-continuation base stride $S_0 = 8$; the final chunk is always processed. With on-demand streaming enabled, preprocessed frames remain on the CPU until their chunks are selected for GPU processing. Efficiency profiling is enabled only for efficiency tests and disabled during accuracy evaluation.

A.4 More Experiments and Visualization Results

Table 7 expands the main comparison in Table 1 by adding an FA-only intermediate result for each host. It retains the short, medium, and long VideoMME and LongVideoBench subsets, MLVU, and the overall average, allowing the contributions of FA and the subsequent addition of EA to be inspected across video lengths.

Separating EA and FA. The combined FA+EA configuration improves the reported Average over its corresponding host for every host/backbone pair in Table 7, including FlashVID and FlexMem on both LLaVA-Video-7B and LLaVA-OneVision-7B. FA alone accounts for part of these gains; adding

EA is not uniformly monotonic. For example, the FlashVID average on LLaVA-Video-7B changes from 62.0 (host) to 62.4 (+FA) and 62.3 (+FA&EA). Per-duration results are likewise mixed: for that host, full integration improves VideoMME long from 53.4 to 54.7 but reduces LongVideoBench long from 53.0 to 51.8. The breakdown therefore documents the component-level trade-off rather than claiming that each added module improves every score.

Table 8: **Efficiency comparison on LLaVA-OneVision-7B.** Blue and red show gains and drops over the ChunkMem. GPU peak memory is PyTorch peak allocated memory.

Method	Ratio	Vision Enc. Lat. (s) ↓	Prefill Lat. (s) ↓	Memory Lat. (ms) ↓	Generate Lat. (s) ↓	GPU Peak Mem. (MB) ↓
ChunkMem Baseline	100%	*3.7*	*31.4*	*486.4*	*36.1*	*23627.1*
VisionZip CVPR'25	25%	3.8 ↑2.7%	7.4 ↓76.5%	414.9 ↓14.7%	12.3 ↓65.9%	19806.3 ↓16.2%
+ Evi-Prefill Ours		**2.3** ↓36.8%	**4.5** ↓85.8%	**254.1** ↓47.7%	**7.6** ↓79.1%	**19067.2** ↓19.3%
FlashVID ICLR'26	25%	3.7 ↑0.1%	7.5 ↓76.1%	425.2 ↓12.6%	13.6 ↓62.4%	19816.7 ↓16.1%
+ Evi-Prefill Ours		**2.0** ↓45.3%	**4.0** ↓87.3%	**220.9** ↓54.6%	**7.4** ↓79.7%	**18995.1** ↓19.6%
FlexMem CVPR'26	15%	3.8 ↑1.4%	17.0 ↓45.8%	730.9 ↑50.3%	21.8 ↓39.8%	19775.5 ↓16.3%
+ Evi-Prefill Ours		**2.5** ↓33.4%	**11.2** ↓64.2%	**283.4** ↓41.7%	**14.3** ↓60.5%	**17988.6** ↓23.9%

Efficiency on LLaVA-OneVision-7B. Table 8 complements the LLaVA-Video efficiency comparison with the same types of latency and peak-memory measurements on LLaVA-OneVision-7B. Relative to its FlashVID host, the integrated configuration reduces vision encoding from 3.7 to 2.0 s, prefilling from 7.5 to 4.0 s, and end-to-end generation from 13.6 to 7.4 s; GPU peak allocation decreases from 19,816.7 to 18,995.1 MB. Relative to FlexMem, the corresponding values change from 3.8 to 2.5 s, 17.0 to 11.2 s, and 21.8 to 14.3 s, with GPU peak allocation falling from 19775.5 to 17988.6 MB. The blue/red percentages in the table compare with *ChunkMem*, whereas these paired comparisons are against each compressed host.

Qualitative examples for FlashVID and VisionZip. As shown in Fig. 9, the orange outline marks the question-relevant visual event, while the timelines distinguish processed, filtered, and skipped chunks and show the dense-to-sparse transition. In both examples, the integrated method preserves the host's correct answer: the processed-chunk count changes from 124/124 to 76/118 for FlashVID and from 128/128 to 71/101 for VisionZip, while the reported generation latency falls from 18.2 to 11.1 s and from 17.0 to 9.4 s, respectively. The denominators differ between host and integrated runs because Frame Allocation filters the input chunk stream before Evidence-aware Allocation selects which remaining chunks to process. These examples illustrate evidence retention under reduced computation.

No-switch and late-switch examples. Fig. 10 complements the earlier cases by showing that Evi-Prefill need not switch early to reduce computation. In the FlashVID example (c), Frame Allocation reduces the input from 56 to 22 chunks; all 22 filtered chunks are processed because EA does not trigger. The host predicts "red," whereas the integrated run predicts the ground-truth "white," with generation latency decreasing from 8.0 to 3.2 s. In the FlexMem example (d), Frame Allocation reduces the input from 128 to 113 chunks. EA then switches only after a substantial dense prefix, processing 74 of the 113 filtered chunks; both runs predict the correct answer "black," while generation latency decreases from 27.9 to 15.7 s. The contrast illustrates that the policy can retain dense processing until its switch criterion is met, and that frame-level savings remain available even when EA does not switch.

A.5 Future Directions

Advantages and contributions. The central insight of Evi-Prefill is that the computational budget for a long video need not be fixed before inference: the usefulness of later observations depends on both the query and the evidence acquired. Our framework turns this insight into two complementary decisions. FA removes structurally repetitive frames before expensive visual encoding, while EA uses a model-internal signal available during chunk prefilling to adapt the density of subsequent processing. This moves beyond compressing incoming chunks to deciding *which* visual observations merit computation in the first place. Because the design is training-free and leaves the host's token compression and memory retrieval intact, it complements different efficient LVLM backbones. More

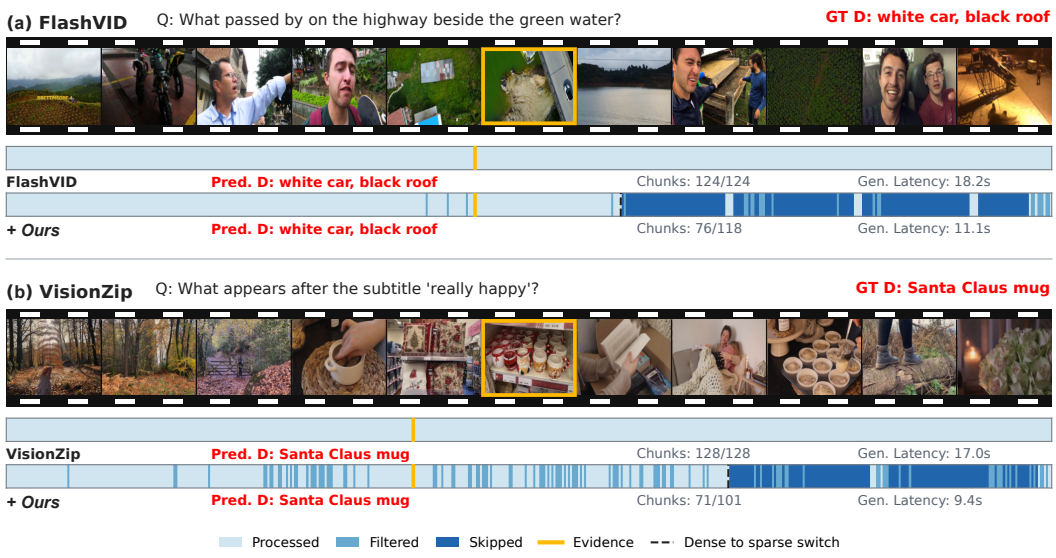

Figure 9: **Additional qualitative examples with FlashVID and VisionZip.** Each example shows the question, sampled video frames, the ground-truth answer, host and integrated predictions, and chunk-processing timelines. Orange marks the relevant event; light blue denotes processed frames, medium blue filtered frames, dark blue skipped frames, and the dashed marker indicates the transition to sparse continuation. Both integrated runs retain the correct answer with fewer processed chunks and lower reported generation latency.

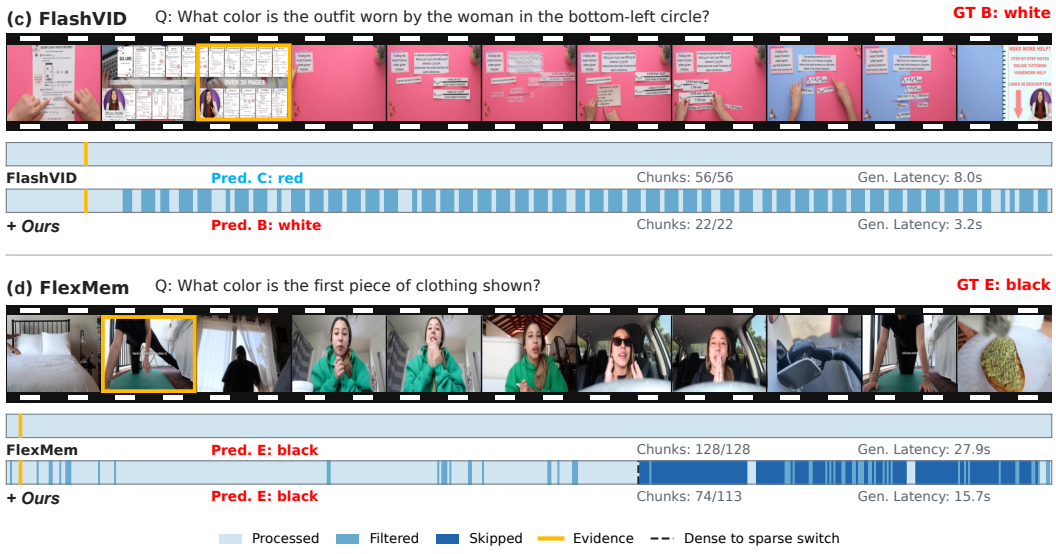

Figure 10: **Additional qualitative examples with FlashVID and FlexMem.**

broadly, it suggests treating long-video understanding as sequential evidence acquisition, with model feedback guiding computation rather than imposing a uniform schedule over time.

Limitations and future directions. NCM is a heuristic proxy derived from LLM self-attention, not ground-truth evidence: distracting content may attract high attention and delay the dense-to-sparse switch. Global reasoning that requires evidence throughout a video also inherently limits the achievable speedup, even with adaptive sparse continuation. FA relies on low-resolution grayscale SSIM

and may overlook small objects or subtle changes. Retaining the first frame of each original chunk during thresholding mitigates this risk, but subsequent frame-count alignment may resample it, and fine-grained evidence is not guaranteed to survive. Moreover, EA switches only once and cannot return to dense processing when later content becomes informative. Future work could explore position-aware guidance signals and allocation policies that adapt within a fixed budget.

Algorithm 1 Evi-Prefill: overall chunk-stream inference

Require: Raw video V_{raw}, query Q, frame budget $N_{\max}$, sampling rate f, chunk size c
Thumbnail size r, SSIM threshold τ_s; EA parameters $(\mathcal{L}_e, \rho, P, \tau_e, S_0)$
Ensure: Answer $\widehat{y}$ and processed chunk set Ω

$\widetilde{V} \leftarrow \text{SampleAndDecodeOnCPU}(V_{\text{raw}}, f, N_{\max}, c)$
$(\widehat{V}, \widehat{\mathcal{C}}_{1:\widehat{B}}) \leftarrow \text{FA}(\widetilde{V}; c, r, \tau_s)$
$\theta_e \leftarrow (\mathcal{L}_e, \rho, P, \tau_e, S_0)$
$(\Omega, M) \leftarrow \text{EA}(\widehat{\mathcal{C}}_{1:\widehat{B}}, Q; \theta_e)$
$\widehat{M} \leftarrow \mathcal{R}(M, Q)$; $\widehat{y} \leftarrow \mathcal{D}(Q, \widehat{M})$
return $(\widehat{y}, \Omega)$

A.6 Algorithm Details of Evi-Prefill

The three algorithms in this section follow the inference order: Algorithm 1 summarizes the full pipeline, Algorithm 2 specifies SSIM-based Frame Allocation (FA), and Algorithm 3 specifies dense-to-sparse Evidence-aware Allocation (EA). They apply to each chunk-stream host; VisualTokens includes vision encoding, projection, and optional pre-LLM token compression, whereas intra-LLM compression is performed within $\mathcal{F}_{\text{pre}}$.

Notation and implementation correspondence. The sampled $\widetilde{V}$ in Algorithms 1 and 2 corresponds to V in Sec. 3.1. The SSIM threshold pass retains the first frame of each original chunk and compares subsequent frames with the most recently retained frame *within that chunk* (Eq. 7). $\text{AlignCount}(n, c, N)$ keeps n when divisible by c, otherwise rounds up to a multiple of c if this does not exceed N and rounds down when it does; when $N < c$, it returns N. UniformAdjust then uniformly fills from rejected sampled frames or subsamples threshold-retained frames. Consequently, the original first-frame guarantee applies to the threshold pass, not necessarily to the final $\widehat{V}$; an incomplete final chunk can be padded internally for model execution. The optional `FILTER_KEEP_RATIO` branch is disabled here.

Algorithm 2 Frame Allocation: chunk-wise SSIM filtering

Require: Sampled frames $\widetilde{V} = \{x_i\}_{i=1}^{N}$; c, r, τ_s
Ensure: Filtered frames $\widehat{V}$, chunks $\widehat{\mathcal{C}}_{1:\widehat{B}}$

$\mathcal{C}_{1:B} \leftarrow \text{Partition}(\widetilde{V}, c)$
$g_i \leftarrow G_r(x_i)$ for all i; $K_{\tau_s} \leftarrow \emptyset$
for all $\mathcal{C}_b$ in temporal order **do**
  $a \leftarrow$ first index in $\mathcal{C}_b$
  $K_{\tau_s} \leftarrow K_{\tau_s} \cup \{a\}$
  for all i after the first index in $\mathcal{C}_b$ **do**
    if $\text{SSIM}(g_a, g_i) \le \tau_s$ **then**
      $K_{\tau_s} \leftarrow K_{\tau_s} \cup \{i\}$; $a \leftarrow i$
    end if
  end for
end for
$\widehat{V}_{\tau_s} \leftarrow \{x_i : i \in K_{\tau_s}\}$
$n^\star \leftarrow \text{AlignCount}(|K_{\tau_s}|, c, N)$
$K \leftarrow \text{UniformAdjust}(K_{\tau_s}, \{1, \ldots, N\}, n^\star)$
$\widehat{V} \leftarrow \{x_i : i \in \text{sort}(K)\}$
$\widehat{\mathcal{C}}_{1:\widehat{B}} \leftarrow \text{Partition}(\widehat{V}, c)$
return $(\widehat{V}, \widehat{\mathcal{C}}_{1:\widehat{B}})$

For EA, $R_{<b}$ denotes the recent context $(R_{b-1}, \ldots, R_{b-k})$, and A_b is query-to-current-visual-token attention obtained from the same prefill pass that writes memory. $\text{Mass}(A_b, \mathcal{Q}_e, \mathcal{L}_e, Z_b)$ implements

Algorithm 3 Evidence-aware Allocation: dense-to-sparse chunk processing

Require: Filtered CPU chunks $\widehat{\mathcal{C}}_{1:\widehat{B}}$, query Q
Last-layer set $\mathcal{L}_e$; parameters ρ, P, τ_e, S_0
Ensure: Processed chunk indices Ω and visual memory M

$\mathcal{Q}_e \leftarrow \mathrm{ExitQueryTokens}(Q)$
$M \leftarrow M_0$; $\Omega \leftarrow \emptyset$; $D \leftarrow 0$; mode $\leftarrow$ dense
$B_{\min} \leftarrow \min\{\widehat{B}, \max(P, \lceil \rho \widehat{B} \rceil)\}$
$S_{\min} \leftarrow \max(1, \lfloor S_0/2 \rfloor)$; $S_{\max} \leftarrow 2S_0$
for $b = 1, \ldots, \widehat{B}$ **do**
  if mode $=$ sparse $\wedge\ b < \widehat{B} \wedge b \not\equiv b_s \pmod{S^\star}$ **then**
    continue ▷ before GPU transfer
  end if
  $Z_b \leftarrow \mathrm{VisualTokens}(\mathrm{H2D}(\widehat{\mathcal{C}}_b))$
  $(H_b, A_b) \leftarrow \mathcal{F}_{\mathrm{pre}}(Q, Z_b, R_{<b}, M)$
  $M \leftarrow \mathcal{U}(M, H_b)$; $\Omega \leftarrow \Omega \cup \{b\}$
  if mode $=$ dense **then**
    $\alpha_b \leftarrow \mathrm{Mass}(A_b, \mathcal{Q}_e, \mathcal{L}_e, Z_b)$
    $D \leftarrow D + \alpha_b$; $m_b \leftarrow 1$ if $D \leq 0$, else $\alpha_b/(D + \epsilon)$
    if $B_{\min} \leq b < \widehat{B} \wedge \max_{b-P<i\leq b} m_i < \tau_e$ **then**
      $b_s \leftarrow b$; $L \leftarrow \widehat{B} - b_s$
      $s \leftarrow \mathrm{round}(S_0 b_s / L)$
      $S^\star \leftarrow \min\{\lceil L/2 \rceil, \mathrm{clip}(s; S_{\min}, S_{\max})\}$
      mode $\leftarrow$ sparse
    end if
  end if
end for
return $(\Omega, \mathrm{FinalizeVisualMemory}(M))$

Eq. 3; $\mathcal{L}_e$ is selected by `ATTENTION_LAST_K`, and the default $\mathcal{Q}_e$ contains the question and answer options. The switch test is applied after chunk b updates memory (Eq. 4); if cumulative attention is zero, m_b is set to 1 to avoid a spurious switch. If the test never fires, all filtered chunks are processed. We use $\mathrm{round}(x) = \lfloor x + 1/2 \rfloor$ for $x \geq 0$. The final chunk is always processed as in Eq. 6. With `ONLINE_FRAME_STREAM=True`, skipped chunks are not transferred to the GPU, encoded, prefilled, or written to memory; CPU video decoding and FA have already taken place.