SAVE: LESS IS MORE IN AGENTIC VIDEO REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic video-reasoning frameworks iteratively explore a video, choosing what to observe based on the query and evidence gathered so far. Existing methods are typically evaluated with different backbones, preprocessing pipelines, and computational budgets, making it difficult to determine which components actually drive performance, how much they cost, and whether agentic inference is preferable to native video reasoning in the first place. We conduct the broadest controlled evaluation to date of a diverse set of recent agentic video-reasoning systems across two strong contemporary backbones, with explicit token and monetary-cost accounting. Across Gemini 3 Flash and Gemma 4, most existing agentic frameworks *fail* to consistently improve over native video inference and often increase computational cost. We investigate two primary questions: (1) How much agentic framework complexity is necessary for effective video reasoning? (2) When, if at all, is agentic video reasoning preferable to just using the backbone directly? To answer the first question, we introduce a deliberately minimal Active Exploration Baseline and show that it remains competitive with substantially more elaborate agents. We then propose targeted extensions to this baseline, yielding SAVE (Simple Agentic Video Explorer). SAVE outperforms all evaluated prior agentic methods in five of six backbone–benchmark settings. On Minerva, SAVE reaches 66.4% accuracy with Gemini 3 Flash, comparable to 64.8% for native inference, while reducing inference cost by 60%, and exceeding the highest-accuracy evaluated prior agentic baseline by 9.9 percentage points with 79% lower cost. Motivated by this fact, we ask when agents are preferable to simply using the underlying backbone. Our results indicate that the advantage of agentic exploration grows when dense processing would otherwise represent answer-relevant events too sparsely, particularly for long videos and temporally concentrated evidence. Overall, our results suggest that agentic video reasoning should favor simple frameworks over increasingly elaborate pipelines and focus primarily on long-form video, where selective evidence acquisition offers the clearest advantage over native inference.

1 INTRODUCTION

Long-form video reasoning remains a fundamental challenge for modern vision-language models because it requires processing large amounts of temporally structured information, performing complex visual reasoning, and integrating evidence over long time horizons. In videos, the observations needed to answer a question may be distributed across thousands of frames, creating a tension between reasoning quality and computational efficiency: densely processing the entire video is expensive, while aggressive downsampling or compression risks discarding critical evidence. Recent multimodal large language models (MLLMs), including GPT-4o (OpenAI, 2024), and Gemini 3 (Google DeepMind, 2025), have substantially improved video understanding, but efficient and accurate reasoning remains challenging.

A growing body of work addresses this problem through agentic video reasoning, where models iteratively select observations conditioned on the query and previously gathered evidence (Zuo et al., 2025; Pang & Wang, 2025; Wang et al., 2025c; Zhi et al., 2025). While previous work asks the question "How do we build better video agents?", we ask two complementary questions: (1) How much agentic machinery is actually necessary? and (2) When, if at all, is agentic inference preferable to simply using the underlying video backbone directly? Existing methods are typically evaluated with different multimodal backbones, computational budgets, preprocessing pipelines, and

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.

benchmark settings. This makes it difficult to determine which components of the agentic framework actually drive performance, what constitutes an effective video agent, and whether agentic inference is preferable to native video reasoning in the first place.

To address those questions, we conduct, to the best of our knowledge, the most comprehensive controlled evaluation to date of contemporary agentic video reasoning frameworks under shared frontier backbones. We systematically evaluate both Gemini 3 Flash (Google DeepMind, 2025) and the open-weight Gemma 4 (Gemma Team, 2026), adapting a diverse set of recent methods to use the same underlying backbone, evaluating them on a common suite of video reasoning benchmarks, and explicitly accounting for computational cost. Surprisingly, we find that when paired with strong multimodal backbones, most existing agentic frameworks *fail* to consistently outperform native video inference and frequently incur substantially greater computational cost. We observe this trend across multiple datasets and agentic formulations.

Despite their differences, most video agents share a common structure: they acquire partial observations, update a reasoning state, and decide what to inspect next. We capture this structure with a simple abstraction inspired by partially observable Markov decision processes (POMDPs). To answer the first question, we instantiate it with a deliberately minimal baseline in which the backbone itself selects individual frames to inspect, without specialized retrieval, hierarchical memory, or video-wide preprocessing. Despite its simplicity, this baseline remains competitive with substantially more elaborate systems.

Building on this observation, we introduce *SAVE (Simple Agentic Video Explorer)*, which improves over the minimal baseline using targeted adjustments. SAVE preserves the same simple interaction loop, but lets the backbone query temporal ranges and adaptively allocates temporal and visual resolution within them. Across three benchmarks and two backbones, SAVE achieves the highest performance among evaluated prior agentic methods in five of six settings, while generally using substantially fewer resources. On Minerva, for example, SAVE reaches 66.4% with Gemini 3 Flash, comparable to 64.8% for native inference and significantly higher than 56.5% for VideoLucy, while reducing monetary inference cost by approximately 60% and 79%, respectively.

Finally, we ask when agentic exploration helps. On Minerva, SAVE's advantage over a token-matched native baseline grows with video duration. A controlled synthetic experiment further shows that adaptive inspection is most useful when the temporal pattern needed to answer a question occupies only a small fraction of the video. Together, our results suggest a simple principle: agentic exploration is most useful when uniform processing would represent answer-relevant evidence too sparsely. Agentic video reasoning appears to be most promising in the long-form regime, where selective evidence acquisition provides the clearest advantage over dense native processing; for shorter videos, the additional overhead of agentic exploration may offer limited benefit.

To summarize, our contributions are threefold: (1) We perform, to the best of our knowledge, the most comprehensive controlled evaluation to date of agentic video reasoning methods under shared frontier backbones with explicit computational accounting, showing that most evaluated frameworks fail to improve over native inference while often substantially increasing cost. (2) We introduce a deliberately minimal Active Exploration Baseline, and build on this baseline with SAVE. Despite its simplicity, the baseline remains competitive with substantially more elaborate agents, while SAVE outperforms all prior agentic approaches in most settings at substantially lower computational cost. (3) We characterize the regime in which agentic exploration is most useful, namely when uniform processing would represent answer-relevant evidence too sparsely.

2 Related Work

MLLMs for video reasoning. Early video MLLMs such as Video-ChatGPT (Maaz et al., 2024), VideoChat (Li et al., 2023), and Video-LLaVA (Lin et al., 2024; Zhu et al., 2024) established the ability of language models to reason over sequential visual inputs. More recent systems, including proprietary models such as Gemini 3 (Google DeepMind, 2025), open-weight models such as Gemma 3 (Gemma Team, 2025), and other recent video MLLMs (Wang et al., 2025b; Zhang et al., 2023; Cheng et al., 2024; Zhang et al., 2025a; Bai et al., 2025), have substantially improved video understanding and reasoning capabilities. Despite these advances, video reasoning remains constrained by a fundamental trade-off between computational cost and information retention. Processing dense

visual inputs preserves fine-grained evidence but incurs substantial memory and compute costs, while uniform downsampling, token compression (Bolya et al., 2023), or pruning (Chen et al., 2024) reduce context size at the risk of discarding question-relevant information. This limitation motivates explicit, query-dependent frame-selection methods.

Frame selection. Frame-selection methods typically operate on top of a vision-capable MLLM backbone and differ primarily in how directly that backbone controls evidence acquisition. We call methods *agentic* when the backbone adapts the high-level search policy based on intermediate observations. Non-agentic methods instead follow a predefined selection procedure before the final reasoning step. We refer to the former as backbone-driven *active exploration* and note that this distinction is a continuum rather than a strict dichotomy.

Non-agentic approaches include T* (Ye et al., 2025), AKS (Tang et al., 2025), and FOCUS (Zhu et al., 2026). Their main advantage is that the expensive final backbone typically processes the selected evidence only once, while frame selection is delegated to predefined and often complex pipelines built around lighter auxiliary models such as CLIP (Radford et al., 2021). While this makes keyframe selection efficient, it also limits the selector to the representations and heuristics encoded by these auxiliary components. As a remedy, recent approaches such as TCoT (Arnab et al., 2025) use the VLM itself in the frame selection pipeline at the cost of using more computational resources.

Agentic approaches have been gaining popularity as an alternative to these fixed selection pipelines. Some first employ an initial, question-agnostic preprocessing step, e.g., to construct a searchable index or subject registry of the entire video. VideoLucy (Zuo et al., 2025), DeepVideoDiscovery (Zhang et al., 2025b), VideoAgent2 (Zhi et al., 2025) and MR Video (Pang & Wang, 2025) follow this paradigm. This step effectively narrows the temporal search space, providing significant benefits for long-form videos that require solving "needle in a haystack" localization problems. However preprocessing is computationally expensive, and because it's performed independently of the question, it risks omitting critical fine-grained details, potentially misguiding the agent's subsequent search. Others rely more on direct query-conditioned exploration. Building on early programmatic precursors (Surís et al., 2023), methods such as VideoAgent (Wang et al., 2024), VideoCuriousAgent (Yang et al., 2025c), VideoTree (Wang et al., 2025c), VideoARM (Yin et al., 2026), and others (Zou et al., 2026a; Chen et al., 2025; Wang et al., 2026), perform actions on the raw video conditioned simultaneously on the query and the previously gathered evidence. VideoZoomer (Ding et al., 2026) is conceptually closely related to our work, as it likewise performs iterative coarse-to-fine temporal exploration; however, it learns a dedicated exploration policy through supervised fine-tuning and reinforcement learning.

Across frame-selection paradigms, we observe a trend in the recent literature toward increasing complexity within the framework itself. Compared with the relatively simple interaction loop of VideoAgent (Wang et al., 2024), DeepVideoDiscovery (Zhang et al., 2025b) first constructs a multi-granular video database and exposes several specialized tools, including global browsing, clip-level search, and frame inspection, which the agent dynamically composes during inference. VideoLucy (Zuo et al., 2025) goes further by maintaining hierarchical coarse-to-fine memories and coordinating multiple specialized agents for localization, instruction generation, captioning, and answering within an iterative backtracking procedure.

Prior work has primarily asked how to build better video agents. However, existing frameworks differ substantially in their formulation, inference budgets, preprocessing pipelines, and underlying backbones, making it difficult to isolate where their gains come from. As a result, two complementary questions remain largely unresolved: (1) How much additional machinery is actually necessary for effective video reasoning? and (2) When does agentic inference provide benefits over using the underlying backbone directly? The focus of our work is to disentangle these factors and answer both questions in a controlled setting.

3 Active Exploration Framework

In this section, we unify various agentic formulations under a single framework. Building upon this foundation, we first establish an Active Exploration Baseline as a minimal instantiation of this framework. We then introduce our proposed approach, SAVE, which addresses the weaknesses of the minimal baseline.

3.1 PROBLEM FORMALIZATION

We observe that most existing agentic approaches can be viewed as instances of a unified framework. To identify the common structure behind those approaches, we present a mathematical formalization, drawing inspiration from partially observable Markov decision processes (POMDPs) (Kaelbling et al., 1998). To demonstrate its generality, this formulation is applied to popular contemporary methods in Appendix B.

Specifically, task-conditioned video inference is modeled as *active exploration*. A problem instance is modeled as E, q, y where $E \in \mathcal{E}$ is the environment (e.g., a video), $q \in \mathcal{Q}$ is the user query, and $y \in \mathcal{Y}$ is the ground-truth answer. In the video reasoning case, $E = V = \{(x_i, t_i)\}_{i=1}^{N}$, where x_i denotes the i-th frame and t_i its timestamp.

Actions and Observations. We define an observation space $\mathcal{O}$ (e.g., frames, crops, text), an action space $\mathcal{A} = \mathcal{A}_{\text{query}} \cup \{\text{STOP}\}$, and an intermediate output (state) space $\mathcal{S}$. At each step the system can observe the environment through a deterministic operator: $O : \mathcal{A}_{\text{query}} \times \mathcal{E} \to \mathcal{O}$.

Internal state. We call s_t the agent state and a_t the action planned at time t. At timestep $t+1$, given the agent state for time t and a_t, the system executes the action $o_{t+1} = O(a_t, E)$, and computes: $s_{t+1} = F(q, s_t, o_{t+1}) \in \mathcal{S}$. The state update F is defined as: $F : \mathcal{Q} \times \mathcal{S} \times \mathcal{O} \to \mathcal{S}$, F aggregates the query q and the evidence collected so far (captured by s_t) together with the newly acquired observation o_{t+1} into an intermediate representation s_{t+1} that is used both for action selection and subsequent predictions. No assumptions are made about the internal state in terms of size and scope; s_{t+1} may also include the entire history of interactions.

Initialization. We define time as starting at $t = 1$, before any query action has been executed. The initial state is produced by $s_1 = F_{\text{init}}(q, E)$. This general form permits both query-dependent initialization and question-agnostic video preprocessing; methods without video-wide preprocessing may simply ignore E.

Policy and termination. The next action, $a_{t+1} \in \mathcal{A}$, is sampled from a policy $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$. If $a_{t+1} \in \mathcal{A}_{\text{query}}$, the process continues with a_{t+1} along with s_{t+1} being passed to the next iteration. If $a_{t+1} = \text{STOP}$, the interaction terminates at time $t+1$. Upon termination, the system outputs a final answer using a prediction head: $\hat{y} = g(s_{t+1}) \in \mathcal{Y}$.

3.2 ACTIVE EXPLORATION BASELINE

We design the baseline agent to be one of the simplest possible instantiations of our agentic framework. The agent operates within an action space $\mathcal{A} = \mathcal{A}_{\text{query}} \cup \{\text{STOP}\}$, where $\mathcal{A}_{\text{query}} = \{\text{QUERY}\}$. Initially, $\mathcal{F}_{\text{init}}$ just proposes a first action based on the question (without an expensive video-wide preprocessing step). At each iteration, it selects either STOP (terminating exploration to provide the final answer) or QUERY(N), where n is a valid frame index. For a given query, the observation operator $\mathcal{O}$ simply extracts the requested frame along with the surrounding transcript obtained using Whisper (Radford et al., 2023). The state update F invokes the backbone with the extracted RGB frame and transcript as well as the question and interaction history. In F, the backbone outputs its reasoning along with the next action, both of which are appended to the interaction history. To prevent context overflow, this history is periodically compressed into a concise text summary using the backbone model. Notably, unlike many previous approaches, we do not temporally downsample the video before exploration; the agent retains access to the full original video and may query any frame at native temporal resolution. The full application of this approach to the above-defined formalism can be found in Appendix B.1.

3.3 SAVE (SIMPLE AGENTIC VIDEO EXPLORER)

While the simple baseline queries individual frames, SAVE expands the query action space to $\mathcal{A}_{\text{query}} = \{\text{QUERYRANGE}(s, e)\}$, where s and e denote the start and end frame indices of a requested temporal interval. This allows the agent to request temporally contiguous regions of the video rather than isolated frames. We found that the lack of temporal context is a primary bottleneck of the minimal baseline, motivating this extension. Given the requested range $[s, e]$, the observation operator applies scene detection using `PySceneDetect` (Castellano) and returns the middle frame of each detected scene segment. The scene-detection granularity is selected automatically as a function of the queried range length. If the initial scene detection threshold yields too few scene segments,

we progressively decrease the detection threshold until sufficient temporal coverage is obtained. If scene detection still fails to produce enough segments, we supplement the middle frames of detected scenes with uniformly sampled ones over the requested interval. SAVE additionally adapts the visual resolution of returned observations to the queried temporal scale: narrow ranges are processed at higher resolution to preserve fine-grained visual details, while broader ranges use default-resolution observations. This allows the agent to trade temporal coverage against visual detail without explicitly selecting these parameters itself. Additional settings are detailed in Appendix A.2. The full agent prompts and the summarized Algorithm 1 are provided in Appendix E.

4 EXPERIMENTS

To the best of our knowledge, our study provides the most comprehensive controlled evaluation to date of contemporary agentic video reasoning methods under shared frontier backbones. We evaluate a diverse set of frameworks using both Gemini 3 (Google DeepMind, 2025) and Gemma 4 (Gemma Team, 2026), while explicitly accounting for computational efficiency. By holding the principal multimodal backbone fixed within each comparison, this evaluation substantially reduces confounding from differences in underlying model capability and enables a more direct assessment of the accuracy–efficiency trade-offs of different exploration strategies.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate on three complementary benchmarks: **Minerva** (Nagrani et al., 2025), which evaluates multi-step reasoning over videos ranging from 2 minutes to over 1 hour; **EgoSchema** (Mangalam et al., 2023), which evaluates temporal, causal, and commonsense reasoning over three-minute egocentric videos; and **VSIBench** (Yang et al., 2025a), which evaluates visual-spatial reasoning. Due to the size of the VSIBench dataset we use the subsampling strategy from the original paper to obtain *VSIBench (tiny)*, which contains 400 samples in total, 50 from each question type.

Baselines. We run our evaluation on two specific backbones: Gemini 3 Flash (Google DeepMind, 2025) and Gemma 4 (Gemma Team, 2026). Our proposed method, SAVE, and the simple active exploration baseline are compared against the respective zero-shot VLM baseline and several popular agentic models: VideoAgent (Wang et al., 2024), VideoTree (Wang et al., 2025c), VideoCuriousAgent (Yang et al., 2025c), VideoLucy (Zuo et al., 2025), DeepVideoDiscovery (Zhang et al., 2025b) and A.I.R. (Zou et al., 2026a). We select these specific baselines because they represent either current state-of-the-art models or conceptually significant exploration paradigms. Additionally, to verify that SAVE's multi-step exploration paradigm provides benefits beyond one-shot query-conditioned localization, we include *SAOS* (Scene-aware one-shot), which restricts SAVE to a single visual query.

Evaluation Details. For each method, we strictly adhere to the official author implementations where available, modifying only the multimodal backbone and the associated prompting logic (e.g. tool calls) to support the standardized backbone. In cases where official code is not publicly accessible, we reimplement the frameworks following the descriptions provided in their respective original publications. For hyperparameters, we follow the original authors' recommendations wherever possible. All hyperparameters can be found in Appendix A.2. We detail and justify all baseline prompts we used in Appendix A.3. For the Gemini 3 native baseline, we utilize the API's default settings along with its built-in audio processing capabilities. We further attempt to include MR Video (Pang & Wang, 2025) and AVP (Wang et al., 2026) in our evaluation. For MR Video we find its preprocessing requirements to be computationally prohibitive, exceeding 3M tokens for the first video in the Minerva dataset. AVP, meanwhile, is not fully backbone-agnostic: its released implementation relies on Gemini's native-video interface to execute temporal sampling and resolution-controlled observations, functionality that is not uniformly available across frontier backbones. Moreover, Gemini's native-video pipeline uses a different visual tokenization scheme from the image-frame inputs used by the other evaluated agents and directly processes the video's audio, whereas our standardized baselines receive Whisper-generated transcript snippets. These differences prevent a directly comparable backbone-standardized evaluation without materially modifying AVP's observation mechanism. We enable thinking for all baselines for evaluation on the Gemini backbone, but disable it when running the Gemma evaluation to test performance on both settings. We always use Gemini-2.5-Flash-Lite for question-agnostic preprocessing steps requiring

Table 1: Performance and inference efficiency across video-reasoning benchmarks and backbones. Each Gemini 3 Flash cell reports *performance / total inference cost*, while each Gemma 4 cell reports *performance / average tokens per sample*. Performance is accuracy for Minerva and EgoSchema and the mean of the eight task-specific metrics for VSIBench. The bottom three rows show our own methods. SAVE achieves the highest performance among evaluated prior agentic methods in five of six backbone–benchmark settings, while generally using substantially fewer resources than more elaborate agentic baselines. Bold indicates the highest performance in each column. See Appendix A.1 for evaluation subsets, metrics, and cost accounting.

	Gemini 3 Flash (*Perf. / Cost*)			**Gemma 4** (*Perf. / Tokens*)		
Model	**Minerva**	**Ego.**	**VSI**	**Minerva**	**Ego.**	**VSI**
Native	64.8 / $80.5	75.4 / $5.75	**59.2** / $10.5	51.6 / 15.9K	72.4 / 16.0K	49.1 / 15.9K
VideoTree	50.4 / $93.4	73.1 / $31.1	45.5 / $20.2	51.6 / 36.4K	71.1 / 34.4K	42.8 / 24.6K
VideoAgent	38.5 / $34.7	64.2 / $4.76	40.6 / $10.6	42.3 / 22.3K	67.7 / 5.1K	44.5 / 26.2K
VideoLucy	56.5 / $154.2	72.7 / $27.4	48.2 / $35.4	58.4 / 72.0K	73.2 / 103.6K	43.8 / 63.4K
VideoCuriousAgent	51.7 / $235.8	72.0 / $51.8	45.6 / $61.2	36.4 / 26.4K	67.7 / 9.9K	41.4 / 24.1K
DeepVideoDiscovery	52.0 / $30.5	69.1 / $11.5	51.9 / $5.21	52.5 / 72.0K	71.5 / 122.2K	49.1 / 34.6K
A.I.R.	55.0 / $39.0	73.9 / $12.8	49.9 / $9.88	51.9 / 22.1K	**74.3** / 23.0K	45.7 / 20.2K
SAOS	57.5 / $13.1	73.1 / $5.46	50.0 / $4.28	52.5 / 8.9K	70.5 / 9.9K	43.5 / 8.6K
Active Exploration	52.8 / $38.1	70.0 / $3.89	47.2 / $9.40	52.6 / 30.4K	65.3 / 10.7K	45.2 / 31.0K
SAVE	**66.4** / $32.0	**76.0** / $6.64	57.1 / $7.80	**62.4** / 19.7K	72.4 / 9.4K	**52.6** / 11.1K

a VLM. To ensure consistent access to speech information, every non-native baseline receives the same Whisper-generated ASR transcript snippets surrounding the frames it requests in all evaluations. Further details on the evaluation protocol can be found in Appendix A.1. To support future research and ensure full reproducibility, we will release all reimplementation code.

4.2 Results

All results are summarized in Table 1. The full breakdown of input and output token counts is provided in Appendix D. Qualitative case studies can be found in Appendix H.

Existing agentic methods do not consistently improve strong video backbones. To our surprise, across both Gemini 3 Flash and Gemma 4, agents do not consistently improve video reasoning performance, irrespective of the evaluated framework. With Gemini 3 Flash, none of the evaluated prior agentic methods outperforms native inference on Minerva, EgoSchema, or VSIBench, while several incur substantially greater inference cost. Remarkably, despite being restricted to a single visual query, SAOS achieves 57.5% on Minerva, outperforming *all evaluated agentic frameworks*, while being significantly cheaper overall than each agentic baseline. The same qualitative trend holds with Gemma 4: methods that improve over native inference often do so at substantially higher token cost. These results suggest that much of the apparent benefit of agentic pipelines diminishes once the underlying multimodal backbone is standardized.

A minimal exploration policy is already competitive. We next investigate what makes an effective video agent: Our deliberately simple Active Exploration Baseline remains competitive with substantially more elaborate methods, reaching 52.8% on Minerva. Its evaluation on Minerva costs less than all other agentic baselines apart from DVD and VideoAgent, both of which underperform this simple baseline. We found this result surprising given the recent trend in video reasoning toward increasingly elaborate agentic frameworks, including hierarchical memories, multi-stage retrieval pipelines, and specialized tool-use mechanisms (Zhang et al., 2025b; Yin et al., 2026; Zuo et al., 2025).

SAVE improves the accuracy–efficiency trade-off. SAVE consistently improves upon the minimal baseline, while still following a deliberately simple design. With Gemini 3 Flash, SAVE outperforms all evaluated prior agentic methods on Minerva, EgoSchema, and VSIBench. On Minerva, SAVE's performance is comparable to native inference at 40% of its total inference cost. Overall SAVE outperforms *every prior agentic framework* in 5 out of 6 backbone–dataset settings. In the Gemini 3 Flash evaluation, SAVE achieves lower inference cost than all but one of the evaluated agentic baselines on each benchmark. The fact that SAVE matches, and in some settings outperforms,

Table 2: Progressive ablation of SAVE. Replacing isolated-frame queries with scene-level range queries yields the largest initial gain across all three benchmarks. High-quality (HQ) observations for narrow ranges further improve Minerva and VSIBench, while adaptive scene detection (AD) gives the strongest performance on these two benchmarks; EgoSchema largely saturates after the scene-level extension. Bold and underline denote the best and second-best values within each dataset, respectively (higher is better for performance; lower is better for token counts).

Dataset	Model	Acc./Ave. (%)	Input Tokens	Output Tokens	Total Tokens
Minerva	Active Exploration Baseline	52.8	**29.4K**	4.2K	**33.6K**
	SAVE w/o HQ and AD	62.7	<u>36.2K</u>	<u>2.5K</u>	<u>38.7K</u>
	SAVE w/o AD	<u>63.3</u>	40.5K	2.8K	43.3K
	SAVE	**66.4**	38.3K	**1.5K**	39.7K
VSIBench (tiny)	Active Exp. Baseline	47.2	**25.8K**	3.5K	**29.3K**
	SAVE w/o HQ and AD	51.3	33.2K	3.0K	36.2K
	SAVE w/o AD	<u>52.5</u>	34.2K	<u>2.7K</u>	36.9K
	SAVE	**57.1**	<u>28.5K</u>	**1.7K**	<u>30.3K</u>
EgoSchema	Active Exploration Baseline	70.0	**9.1K**	1.1K	**10.2K**
	SAVE w/o HQ and AD	**77.0**	<u>16.9K</u>	<u>0.9K</u>	<u>17.8K</u>
	SAVE w/o AD	75.6	17.8K	1.0K	18.8K
	SAVE	<u>76.0</u>	21.6K	**0.9K**	22.5K

the corresponding backbone provides evidence that native inference is not always preferable to a well-designed agentic formulation, which motivates our analysis in Section 5.

Statistical Significance and Comparison to Existing Results. For benchmarks evaluated using per-example accuracy, we assess statistical significance using paired, two-sided tests with Holm correction for multiple comparisons: video-clustered permutation tests for Minerva, where multiple questions may correspond to the same video, and exact McNemar tests for EgoSchema. In particular, SAVE significantly outperforms all agentic baselines on Minerva in both the Gemma and Gemini 3 Flash evaluations. The full results can be found in Appendix F.1. As further validation of our baseline reimplementations, we compare our results against previously reported results from related work in Appendix C.

4.3 Architecture Ablations

To understand the impact of our proposed enhancements, we systematically ablate the components of SAVE, building upon the simple Active Exploration Baseline. The results of this progressive ablation study across all datasets are presented in Table 2.

Starting from the simple baseline, our initial analysis reveals that context breadth is the primary bottleneck. To address this, we transition from querying isolated frames to sampling entire scenes (**SAVE w/o HQ and AD**). In this setup the model chooses the granularity (low, medium, or high), which controls the scene detection thresholds. This substantially improves performance across all benchmarks, indicating that context breadth is a major component of successful video reasoning. Notably, performance on EgoSchema saturates after this improvement, suggesting that this is the major bottleneck on that particular benchmark [1]. Certain tasks, however, require resolving precise visual details where perception becomes the limiting factor. To compensate, we introduce high-quality frames specifically for queries targeting fine temporal granularities (**SAVE w/o AD**). Consistent with our prior observation, performance increases on both Minerva and VSIBench. Finally, we observe that the model occasionally struggles in querying scenes at the right granularity. For example, this occurs in fine-grained motion sequences where standard scene detection fails to isolate meaningful semantic boundaries. To mitigate this, we introduce our adaptive threshold mechanism as described in Section 3.3. Its addition yields our final formulation (**SAVE**), showing the strongest overall performance across the benchmarks.

[1] An explanation for this is that EgoSchema tends to require less attention to small details than Minerva or VSIBench, because broad global context is usually sufficient for a correct answer.

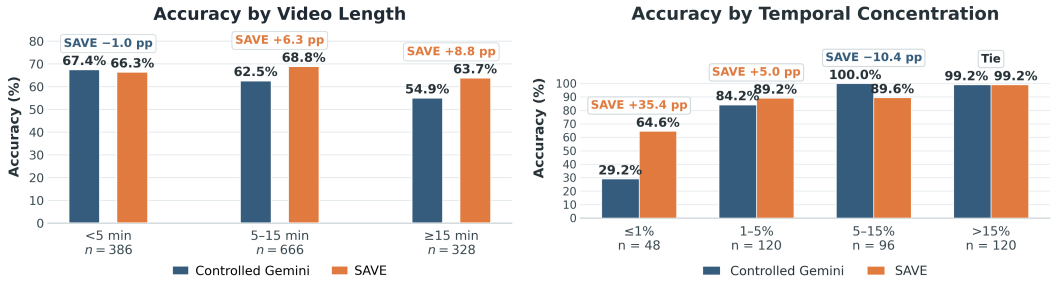


Figure 1: Controlled comparison with Gemini 3 Flash under approximately matched input-token budgets. Left: on Minerva (Nagrani et al., 2025), SAVE is comparable to native inference for videos under 5 minutes, but its advantage grows with video length. Right: on our synthetic benchmark, SAVE provides the largest gain when answer-relevant events occupy only a small fraction of the video; the advantage vanishes as the required temporal pattern becomes broadly represented. Bars show accuracy; n denotes the number of samples in each bin.

5 When does Agentic Video Reasoning Help?

Motivated by the fact that SAVE is competitive with, and in several settings outperforms, native processing, we systematically investigate when the agentic formulation is preferable to native inference and vice-versa. Our results indicate that two key factors are *video length* and *temporal concentration of answer-relevant evidence*.

Video Length. Dense processing remains practical for short videos, leaving limited scope for active exploration to improve accuracy while introducing substantial overhead. Selective inspection becomes increasingly valuable as videos grow longer, because dense processing must distribute computation across the full video and can incur substantial memory and compute costs. In light of this, we hypothesize that under equal token budgets active exploration becomes increasingly useful as the duration of the video grows. We test this hypothesis on Minerva, whose videos range from under two minutes to over an hour. Using Gemini 3 Flash for both approaches, we calibrate the sampling rate of direct full-video inference to approximate SAVE's per-sample input-token consumption while holding the generation configuration fixed. SAVE significantly outperforms Controlled Gemini (66.88% vs. 62.03%; video-clustered permutation test, p=0.0055). As shown in Figure 1, the methods perform comparably on videos shorter than five minutes (66.32% vs. 67.36%; p=0.776). SAVE's advantage then grows to 6.31 percentage points on 5–15 minute videos (68.77% vs. 62.46%; p=0.009) and 8.84 points on videos lasting at least 15 minutes (63.72% vs. 54.88%; p=0.001).

Temporal Concentration of Evidence. Duration alone, however, does not fully determine whether active exploration is beneficial. Even in short videos, answering a question may require resolving a temporal pattern that unfolds over only a small fraction of the video's duration. Under a fixed input-token budget, uniform sampling spreads observations across the entire video and may therefore sample too sparsely to capture such fine-grained temporal structure. An agentic method can instead adapt its sampling granularity, "zooming in" to inspect candidate intervals at higher temporal resolution.

To isolate this effect, we introduce a synthetic video question-answering benchmark with 384 samples and video durations ranging from 30 seconds to 10 minutes. Each video contains a precisely defined answer-relevant event that repeats continuously throughout the video, making localization trivial. This construction allows us to study whether adaptive sampling helps when the temporal resolution required to resolve an event becomes fine relative to the overall video duration, separately from the difficulty of localizing the event itself. Details are provided in Appendix G. We define *temporal concentration* as the fraction of the video occupied by one sufficient event, i.e., the event duration divided by the video duration. Smaller values indicate that a sufficient event unfolds over a smaller fraction of the video and therefore requires finer temporal resolution relative to the overall video duration.

Across all 384 common samples, SAVE and Controlled Gemini achieve 89.32% and 85.94% accuracy, respectively. However, as shown in Figure 1, when an individual event lasts at most 1% of the video,

SAVE significantly outperforms Controlled Gemini by 35.42 percentage points (64.58% vs. 29.17%, $p = 0.0015$). As the event duration increases relative to the video duration, uniform sampling becomes increasingly likely to capture the relevant temporal pattern at sufficient resolution, reducing the benefit of adaptive refinement and making the overhead of the agentic formulation less worthwhile. To verify that SAVE obtains its advantage through iterative adaptive sampling, we also evaluate SAOS on the synthetic dataset. SAOS performs substantially worse overall (46.1% vs. 89.3%). SAVE's lead generally increases as the sequence of events becomes more temporally concentrated: 24.2 percentage points for (>15%), 36.5 for (5–15%), and 69.2 for (1–5%), before narrowing to 39.6 points for ($\leq$ 1%). We also confirm that SAVE's advantage on this benchmark is not attributable to scene detection: At every threshold used by SAVE, PySceneDetect did not detect a single scene transition in any of the synthetic samples.

Implications. Together, these experiments suggest a simple principle: active exploration is most useful when a fixed processing budget would otherwise represent answer-relevant evidence too sparsely. Long-form video is therefore a particularly promising regime, since temporal coverage becomes increasingly expensive while fine-grained events may still need to be resolved. This interpretation is consistent with the direction of recent work: The temporal horizon targeted by agentic methods has shifted toward progressively longer videos. Early approaches such as VideoAgent (Wang et al., 2024) evaluated on minute-scale benchmarks such as NExT-QA (Xiao et al., 2021), while VideoTree (Wang et al., 2025c) already extended evaluation to the long split of Video-MME (Fu et al., 2025), with videos averaging roughly 44 minutes. This shift is also reflected in the benchmark landscape: LVBench (Wang et al., 2025a) explicitly targets extreme long-video understanding, increasing the average video duration to approximately 68 minutes, compared with roughly 10 minutes for earlier long-video benchmarks. More recent systems such as DeepVideoDiscovery (Zhang et al., 2025b) explicitly target information-dense hour-long videos, while the work of Rege et al. (2026) pushes the setting further toward longitudinal video streams spanning multiple days.

6 Limitations

Our study is limited to a finite set of contemporary agentic frameworks, two strong multimodal backbones, and three video-reasoning benchmarks. Standardizing heterogeneous methods requires adapting backbone calls, prompts, and tool interfaces, which may affect their behavior despite our efforts to preserve each method's algorithmic structure. We also do not evaluate on an ultra-long benchmark such as LVBench (Wang et al., 2025a), where expensive video-wide preprocessing may become more advantageous.

Finally, our efficiency analysis primarily measures token consumption and inference cost. These capture the dominant model-side computational expense, but do not fully reflect all factors relevant in practice. We do not include end-to-end latency, as in our evaluation setting it is highly noisy and strongly influenced by external factors unrelated to the method itself, such as public API rate limiting, transient outages, network variability and heterogeneous hardware setups. We therefore view token usage and inference cost as the more reliable basis for controlled comparison.

7 Conclusion

We presented a unified view of agentic video reasoning and conducted a controlled evaluation of contemporary methods under shared frontier backbones. Our results show that increasingly elaborate agentic frameworks do not reliably outperform native video inference, while a minimal backbone-driven exploration strategy remains highly competitive. Building on this observation, SAVE outperforms all evaluated prior agentic frameworks in five of six backbone–benchmark settings. Moreover, our analysis suggests that agentic video reasoning is most useful when answer-relevant evidence requires fine temporal resolution relative to the overall video duration, or when exhaustive native processing is costly. These findings motivate future agentic video reasoning research to focus primarily on long-form video understanding.

AI USE STATEMENT

AI tools were used to support literature research, assist with the implementation of experimental baselines, and check these implementations for faithfulness to the corresponding papers and released code. They were also used for proofreading and improving the clarity of the manuscript, as well as for supporting data analysis. All methodological decisions, experimental design choices, implementation decisions, and interpretations of the results were made and verified by the authors.

REFERENCES

Anurag Arnab, Ahmet Iscen, Mathilde Caron, Alireza Fathi, and Cordelia Schmid. Temporal chain of thought: Long-video understanding by thinking in frames. In *Advances in Neural Information Processing Systems*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, et al. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023.

Brandon Castellano. PySceneDetect. URL https://github.com/Breakthrough/PySceneDetect. Video cut detection and analysis tool.

Boyu Chen, Zhengrong Yue, Siran Chen, Zikang Wang, Yang Liu, Peng Li, and Yali Wang. Lvagent: Long video understanding by multi-round dynamical collaboration of mllm agents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20237–20246, October 2025.

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 *European Conference on Computer Vision (ECCV)*, 2024.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 2024. URL https://arxiv.org/abs/2406.07476.

Yang Ding, Xin Lai, Yizhen Zhang, Wei Li, Ruihang Chu, and Yujiu Yang. Videozoomer: Reinforcement-learned temporal focusing for long video reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 20087–20111, 2026.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118, June 2025.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Gemma Team. Gemma 4 technical report. 2026. doi: 10.48550/arXiv.2607.02770.

Google DeepMind. Gemini 3 flash model card. https://deepmind.google/models/model-cards/gemini-3-flash/, 2025.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Kunchang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

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 *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, 2024.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *Advances in Neural Information Processing Systems*, 2023.

Arsha Nagrani, Sachit Menon, Ahmet Iscen, Shyamal Buch, Ramin Mehran, Nilpa Jha, Anja Hauth, Yukun Zhu, Carl Vondrick, Mikhail Sirotenko, Cordelia Schmid, and Tobias Weyand. Minerva: Evaluating complex video reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/papers/Nagrani_MINERVA_Evaluating_Complex_Video_Reasoning_ICCV_2025_paper.pdf.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Ziqi Pang and Yu-Xiong Wang. Mr. video: Mapreduce as an effective principle for long video understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, 2021.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Aniket Rege, Arka Sadhu, Yuliang Li, Kejie Li, Ramya Korlakai Vinayak, Yuning Chai, Yong Jae Lee, and Hyo Jin Kim. Agentic very long video understanding. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 46575–46602, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2161. URL https://aclanthology.org/2026.acl-long.2161/.

Bytedance Seed. Seed2.0 model card: Towards intelligence frontier for real-world complexity, 2026. URL https://arxiv.org/abs/2607.00248.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, October 2023.

Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29118–29128, June 2025.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, and Jie Tang. Lvbench: An extreme long video understanding benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22958–22967, October 2025a.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision (ECCV)*, 2024.

Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, Min Dou, Kai Chen, Wenhai Wang, Yu Qiao, Yali Wang, and Limin Wang. Internvideo2.5: Empowering video mllms with long and rich context modeling. *arXiv preprint arXiv:2501.12386*, 2025b.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 3272–3283, 2025c.

Ziyang Wang, Honglu Zhou, Shijie Wang, Junnan Li, Caiming Xiong, Silvio Savarese, Mohit Bansal, Michael S. Ryoo, and Juan Carlos Niebles. Active video perception: Iterative evidence seeking for agentic long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9088–9099, June 2026.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9777–9786, June 2021.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10632–10643, June 2025a.

Te Yang, Xiangyu Zhu, Bo Wang, Quan Chen, Peng Jiang, and Zhen Lei. Eea: Exploration-exploitation agent for long video understanding, 2025b. URL https://arxiv.org/abs/2512.03500.

Zeyuan Yang, Delin Chen, Xueyang Yu, Maohao Shen, and Chuang Gan. Vca: Video curious agent for long video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025c. URL https://openaccess.thecvf.com/content/ICCV2025/papers/Yang_VCA_Video_Curious_Agent_for_Long_Video_Understanding_ICCV_2025_paper.pdf.

Jinhui Ye, Zihan Wang, Haosen Sun, Keshigeyan Chandrasegaran, Zane Durante, Cristobal Eyzaguirre, Yonatan Bisk, Juan Carlos Niebles, Ehsan Adeli, Li Fei-Fei, Jiajun Wu, and Manling Li. Re-thinking temporal search for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8579–8591, June 2025.

Yufei Yin, Qianke Meng, Minghao Chen, Jiajun Ding, Zhenwei Shao, and Zhou Yu. Videoarm: Agentic reasoning over hierarchical memory for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24042–24051, June 2026.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhu Chen, and Graham Neubig. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.736. URL https://aclanthology.org/2025.acl-long.736/.

Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, Peng Jin, Wenqi Zhang, Fan Wang, Lidong Bing, and Deli Zhao. Videollama 3: Frontier multimodal foundation models for image and video understanding. *arXiv preprint arXiv:2501.13106*, 2025a. URL https://arxiv.org/abs/2501.13106.

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023. URL https://arxiv.org/abs/2306.02858.

Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Yue Zhao, Ishan Misra, Philipp Krähenbühl, and Rohit Girdhar. Learning video representations from large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6586–6597, June 2023.

Zhuo Zhi, Qiangqiang Wu, Minghe Shen, Wenbo Li, Yinchuan Li, Kun Shao, and Kaiwen Zhou. Videoagent2: Enhancing the llm-based agent system for long-form video understanding by uncertainty-aware cot. *arXiv preprint arXiv:2504.04471*, 2025.

Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, Hongfa Wang, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. In *The Twelfth International Conference on Learning Representations*, 2024.

Zirui Zhu, Hailun Xu, Yang Luo, Yong Liu, Kanchan Sarkar, Zhenheng Yang, and Yang You. Focus: Efficient keyframe selection for long video understanding. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Yuanhao Zou, Shengji Jin, Andong Deng, Youpeng Zhao, Jun Wang, and Chen Chen. A.i.r.: Enabling adaptive, iterative, and reasoning-based frame selection for video question answering. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Zhishan Zou, Guoyan Sun, Zhiwei Wei, Jiancheng Pan, Yujie Li, Mugen Peng, and Wenjia Xu. Self in space: Benchmarking self-awareness and spatial cognition in uav embodied intelligence, 2026b. URL https://arxiv.org/abs/2607.12477.

Jialong Zuo, Yongtai Deng, Lingdong Kong, Jingkang Yang, Rui Jin, Yiwei Zhang, Nong Sang, Liang Pan, Ziwei Liu, and Changxin Gao. Videolucy: Deep memory backtracking for long video understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Implementation details

A.1 Evaluation Protocol

Gemma Evaluation For the Gemma 4 evaluation, which we run locally, we are restricted by the available GPU memory and therefore cannot process an arbitrary number of frames in a single model call. We consequently limit the number of frames provided to Gemma 4 to at most 60 per call for all evaluated methods. For the native Gemma 4 baseline, we uniformly sample 60 frames from the full video. For agentic methods, this same per-call limit is applied to every intermediate reasoning or observation step, ensuring that all methods operate under the same model-level memory constraint. We emphasize that the memory requirements are an explicit disadvantage of native inference.

Evaluation Subsets For all experiments, a small portion of videos in each dataset is either blocked by the Gemini API's safety filters or no longer publicly available. For Minerva specifically a small portion of videos exceed Gemini's context length using the default settings. Consequently, we evaluate on the remaining accessible subsets. Specifically, on Minerva we evaluate 1359 out of 1418 samples, and on EgoSchema we evaluate 491 out of 500 samples from the labeled subset. For VSIBench, due to the high computational overhead of many-agent comparisons, we evaluate 400 samples (taking the first 50 from each question type), following the subsampling strategy used in the original paper Yang et al. (2025a). For the controlled Minerva experiment in Section 5, we evaluate on the 1,380 samples for which both SAVE and the token-matched native baseline produce valid outputs. This subset differs from the 1,359-sample subset used in the main evaluation because the controlled native baseline processes fewer uniformly sampled frames, causing fewer videos to exceed Gemini's context limit. All comparisons within each experiment are performed on the corresponding common valid subset.

Metrics For our evaluation on EgoSchema and Minerva, we report the standard Accuracy metric. For VSIBench, following the original evaluation protocol, we report the average score across its eight question types. Numerical-answer tasks (object counting, object size, room size, and absolute distance) are evaluated using Mean Relative Accuracy (MRA), which measures numerical accuracy across a range of relative-error tolerances. Multiple-choice tasks (relative distance, relative direction, route planning, and appearance order) are evaluated using standard accuracy. The overall VSIBench score is the unweighted average across these eight task-level scores.

Cost Accounting As Gemini 3 Flash is a proprietary model accessed through an API, we use monetary inference cost, based on the provider's API pricing, as the primary efficiency metric. When additionally reporting token counts for Gemini 3 Flash, thinking tokens are included as output tokens. For methods that require question-agnostic preprocessing, the corresponding preprocessing cost is incurred only once per video and amortized across all questions associated with that video. Gemma 4 is an open-weight model that we evaluate locally, so there is no corresponding API-based inference cost; we therefore report total model tokens as the primary efficiency metric. Since Gemma 4 is evaluated without thinking enabled, its output-token counts correspond only to the visible generated output. For readability, we always report per-sample averages for token counts and totals across the full evaluation for monetary inference cost

A.2 Hyperparameters

All Gemini-based baselines used `gemini-3-flash-preview` with Decord frame decoding.

VideoAgent. VideoAgent initially sampled 5 uniformly spaced frames and allowed up to 4 refinement iterations. Additional frames during refinement were selected using CLIP-based frame retrieval. The final answer was taken from the last model response.

VideoTree. We sampled frames at 1 FPS and used EVA-CLIP embeddings with cosine k-means. We set the initial number of clusters to 8, maximum breadth to 32, maximum depth to 3, branch width to 4, and the high-relevance stopping threshold to 4. Gemini calls used temperature 0.0, a thinking-budget cap of 256 tokens, and caption chunks of 10 frames.

VideoCuriousAgent. Each exploration step sampled 4 frames from the selected segment. We used a memory buffer of 16 frames and capped exploration at 5 iterations. Gemini calls used temperature 0.5.

VideoLucy. Coarse memory captioning used `gemini-2.5-flash-lite`; fine/ultra-fine captioning and all reasoning calls used `gemini-3-flash-preview` or `gemma-4-31b-it`. We used a thinking-budget cap of 2048 tokens and at most 5 backtracking iterations. Coarse memory was sampled at 1 FPS and fine/ultra-fine clips at 2 FPS. For Minerva, the coarse temporal scope was set by video duration to 30/60/100/200 seconds for videos shorter than 600/1200/3600 seconds or longer, respectively; the fine scope was 5/5/10/10 seconds. The ultra-fine scope was 1 second.

DeepVideoDiscovery. The video database was built from 10-second clips sampled at 2 FPS. Preprocessing used `gemini-2.5-flash-lite` for clip captioning/registry merging and `gemini-embedding-001` with 3072-dimensional embeddings. The agent used `gemini-3-flash-preview` or `gemma-4-31b-it` with function calling. We capped interaction at 3 iterations, used a global-browse top-k of 300, and allowed frame-inspection calls to include up to 50 frames.

SAVE. SAVE allowed the active vision agent to issue up to 30 visual queries before producing a final answer. Each query specified a frame-index range, and the scene-detection parameters were selected automatically based on the duration of the requested interval. For ranges of at most 10 seconds, we used a target of 6 scenes and `ContentDetector` thresholds of 12, 10, 8, and 6. For ranges of at most 60 seconds, we used a target of 12 scenes and thresholds of 20, 18, 16, 14, 12, and 10. For ranges of at most 8 minutes, we used a target of 25 scenes and thresholds of 30, 25, 20, 18, 16, 14, and 12. For longer ranges, we used a target of 6 scenes and thresholds of 40, 35, 30, 25, 20, 16, 12, and 10. For each query, PySceneDetect progressively reduced the detection threshold until the target number of scenes was reached or all configured thresholds had been evaluated. The environment returned the middle frame of each detected scene and uniformly subsampled the detected scenes to at most $K = 40$ frames per query. If no threshold produced the target number of scenes, SAVE instead returned the target number of uniformly spaced frames over the queried interval: Query ranges of at most $\delta = 10$ seconds used high-resolution visual inputs, whereas longer ranges used medium-resolution inputs. For each returned frame, SAVE additionally provided the local transcript from approximately one second before to one second after the corresponding timestamp. Across iterations, SAVE retained a text-only interaction history containing the queried ranges, relevant frame indices, transcript snippets, and the model's textual observation summaries. The history was compressed every $C = 5$ exploration steps using the backbone with a thinking budget of 2048 tokens, while preserving answer-relevant evidence and temporal search anchors.

A.3 Baseline Prompts

We provide the prompts we use inside each baseline implementation along with any deviations from the original formulations. To ensure consistent access to speech information across methods, we append ASR-generated transcripts to the prompts below if applicable.

A.3.1 VideoTree

Captioning prompt

> You are a precise video captioner. You will receive multiple frames from a single video together with their frame indices.
> **Question context:**
> *[question]*
> You **must** caption every frame with an index in
>
> [*frame indices*]
>
> exactly once. Do not skip or duplicate any frame index.
> For each frame, output one concise sentence describing the visible actions and objects relevant to the question.

> Return only a JSON array containing exactly one object per frame, using the following schema:
>
> `[{"frame_idx": 123, "caption": "..."}, ...]`
>
> Always use exactly the keys `frame_idx` and `caption`. Do not include Markdown, additional keys, or any text outside the JSON array.

The VideoTree repository [2] does not provide a captioning prompt as all the captions come pregenerated from LaViLA Zhao et al. (2023). We therefore introduce the prompt above to obtain one question-conditioned caption for each requested frame.

Relevance Scoring Prompt

> You are presented with a textual description of a video clip consisting of N frame captions sparsely sampled from the video. Your goal is to answer a question about the video. Imagine the visual scene as vividly as possible to improve the accuracy of your response. After selecting an answer, assign a confidence score from 1 to 100, where 1 indicates low confidence and 100 indicates high confidence. Provide a concise, one-sentence explanation for your answer. If uncertain, select the answer that appears most likely to be correct. Additionally, assign a relevance score to each frame caption according to its relevance to answering the question. Each score must be one of $\{1, 2, 3\}$, where 1 indicates low relevance and 3 indicates high relevance. Return exactly one relevance score per frame caption, preserving the order in which the captions are provided. If the question contains answer options, return the corresponding option identifier as the prediction, such as `A`, `B`, `C`, or a numeric identifier. If no explicit answer options are provided, return a short free-form answer. Never return `null`, `unknown`, `N/A`, or `-1`; when the exact answer cannot be determined, provide the most plausible estimate. **Description:** *[frame captions]* `###` **Transcript snippets:** *[transcript]* `###` **Question:** *[question]* `###`
>
> Return only a JSON object in the following format:
>
> `{ "prediction": "...", "explanation": "...", "confidence": 1, "frame_relevance`

The original VideoTree prompt is tailored to EgoSchema: it assumes an egocentric video, uses dataset-specific caption conventions such as #C and #O, and restricts predictions to five lettered answer choices. We adapt the prompt for evaluation across a broader set of benchmarks by removing these dataset-specific assumptions and supporting each benchmark's answer format. For a fair comparison, we also include the same ASR transcript provided to every evaluated method. Finally, because the evaluated backbones did not reliably follow the original free-form response format, we require structured JSON output to ensure robust and consistent parsing. These adaptations preserve the core relevance-scoring task of assigning a score from 1 to 3 to each selected caption.

Final Reasoning Prompt

> You are given a query-adaptive, coarse-to-fine set of frame captions from a long video. Use them to answer the question as accurately as possible.
>
> **Transcript snippets:**
>
> *[transcript]*
>
> **Frame captions in temporal order:**
>
> *[frame captions]*
>
> **Question:**
>
> *[question]*
>
> If the question provides answer options, return the corresponding option identifier as the prediction, such as `A`, `B`, `C`, or a numeric identifier. If the question requires a

[2] https://github.com/Ziyang412/VideoTree

numeric answer and does not provide answer options, return only the number as the prediction, for example `3.5`.

Never return `null`, `unknown`, `N/A`, or `-1`. If the exact answer cannot be determined, provide the most plausible estimate.

Return only a JSON object with the following keys:

```
{
  "prediction": "option identifier or numeric answer",
  "explanation": "brief reasoning",
  "confidence": 1
}
```

The value of `confidence` must be an integer from 1 to 100.

Again we adapt VideoTree's final-reasoning prompt to support multiple benchmarks rather than the EgoSchema-specific five-choice setting. The adapted prompt supports benchmark-specific option identifiers and numerical answers and presents the selected captions in temporal order. For a controlled comparison, each caption is annotated with its source timestamp, matching the temporal information available to the other evaluated methods and supporting questions involving event order, duration, or explicit temporal references. We also include the same ASR transcript supplied to every method. Finally, we require structured JSON output to ensure reliable parsing across model backbones.

A.3.2 VIDEOLUCY

Coarse Captioning Prompt

> Observe and understand the given video carefully. Describe its contents as comprehensively as possible in a smooth and coherent passage. Do not omit any details or prominent information. If the video contains any text, subtitles, text overlays, or voice-overs, explicitly describe their content in detail.

This prompt is identical to the one from the VideoLucy repo[3].

Coarse Answering Prompt

> The following provides a rough description of what is shown in the video during different time periods:
>
> **Video descriptions:**
>
> *[memory]*
>
> Note that since these descriptions are not very complete and detailed, some key information in the video segments of each time period may not all appear in these content descriptions.
>
> This is an academic video understanding benchmark task. The following is the benchmark question to answer:
>
> **Question:**
>
> *[question]*
>
> Please read the given video content descriptions and the question in depth, and determine whether you can accurately answer the given question solely based on the currently provided descriptions.
>
> If you can answer it with absolute confidence, please answer this question and provide the time periods you are referring to. The answer you provide must have completely and absolutely objective support in the video descriptions. Do not make inferences arbitrarily.
>
> If you think the current content descriptions of the video are still insufficient to accurately answer the question, please do not answer it and give me your reason.
>
> Please output in a strictly standardized dictionary format containing four key–value pairs:

[3] https://github.com/worldbench/VideoLucy

- `"Confidence"`: A boolean value. Set it to `True` if you are certain about the answer, and `False` if not.
- `"Answer"`: A string. This string must be enclosed in double quotes. When `"Confidence"` is `True`, fill in the correct answer; when `"Confidence"` is `False`, fill in `"No Answer"`. This must be exactly one number, either the number of the option, if options are given, or the numeric answer to the question, if no options are given.
- `"Time Period"`: A list. When `"Confidence"` is `True`, fill in the list with time periods corresponding to the answer, each in the format of a tuple `(start time, end time)`; when `"Confidence"` is `False`, fill in `"No Time"`.
- `"Reason"`: A string. This string must be enclosed in double quotes. Show me your reasoning about your judgment. You need to ensure and check that your reasoning must be able to absolutely support your answer.

Note that no additional comments should be added within the dictionary. You must note that if an ordinal number appears in the provided question, in the vast majority of cases, you should not simply assume that this ordinal number represents the ordinal of the provided time period. You need to focus on understanding the specific meaning of this ordinal number within the question based on all the content descriptions. You **must** output a valid JSON dictionary.

Return **only** a valid JSON object.

The response must start with `{` and end with `}`.

Do not wrap the JSON in Markdown fences.

Do not add explanations before or after the JSON.

Use double quotes for all keys and string values.

Use JSON booleans `true`/`false`, not Python booleans `True`/`False`.

For `"Time Period"`, use JSON arrays such as `[[120, 150]]`, not tuples such as `[(120, 150)]` or strings such as `"120s-150s"`.

For this prompt, in contrast to the original prompt, we again adjust the answer format to be dataset-agnostic, require a strict JSON output for reliability and introduce the academic benchmark framing to bypass Gemini API filters for certain Minerva questions.

Initial Localization Prompt

The following provides a rough description of what is shown in the video during different time periods:

Video descriptions:

[memory]

Now, a question has been raised regarding this video:

Question:

[question]

Please read the given video content descriptions and the question in depth.

Since most of these descriptions are rather rough and some detailed information is lost, my task is to try my best to find the time periods related to the given question and then provide more detailed descriptions of the video content during these time periods.

In order to assist me in completing this task, your task is to determine, based on the provided rough video descriptions, whether further observing the video content during two time periods would allow me to answer the given question with greater confidence.

If so, identify the time periods related to the question as accurately as possible and provide these relevant time periods so that I can review the corresponding video segments again, obtain more information, and answer the question more accurately.

For example, since they do not require an overall understanding of large video segments, the following questions can be answered more accurately by carefully re-observing two video segments:

(i) What color is Putin's tie between the interview with Antony Blinken and the interview with Marie Yovanovitch?
(ii) How does the goalkeeper prevent Liverpool's shot from scoring at 81:38 in the video?
(iii) Who smashes the magic mirror?

In contrast, because they require an overall understanding of large video segments, it is difficult to answer the following questions more accurately by observing only two video segments:

(i) What happens in the second half of the game?
(ii) What is the video about?
(iii) Which places has the protagonist of this video visited in total?

You should output a strictly standardized dictionary containing exactly three key–value pairs:

- `"Flag"`: A Boolean value. If you are very confident that you can provide time periods according to the requirements above, set it to `True`. Otherwise, set it to `False`.
- `"Time Period"`: A list. If `"Flag"` is `True`, fill the list with the two most relevant time periods, each in the tuple format `(start time, end time)`. If `"Flag"` is `False`, return `"No Time Periods"`.
- `"Reason"`: A string enclosed in double quotation marks. Explain the reasons for the time periods you provided.

No additional comments should be added within the dictionary.

Return **only** a valid JSON object.

The response must start with `{` and end with `}`.

Do not wrap the JSON in Markdown fences.

Do not add explanations before or after the JSON.

Use double quotation marks for all keys and string values.

Use JSON Boolean values `true`/`false`, not Python Boolean values `True`/`False`.

For `"Time Period"`, use JSON arrays such as `[[120, 150]]`, not tuples such as `[(120, 150)]` or strings such as `"120s-150s"`.

We preserve VideoLucy's original localization-and-instruction prompt while stating its existing exclusion of previously searched periods more explicitly, as the API-based model occasionally revisited excluded intervals in preliminary Minerva runs. For cross-dataset evaluation, we additionally embed answer choices in the question and enforce valid JSON outputs for reliable inference through the Gemini API.

Localization and Instruction Agent Prompt

The following provides a rough description of what is shown in the video during different time periods:

Video descriptions:

[memory]

Note that since these descriptions are not very complete or detailed, some key information from the video segments corresponding to each time period may not appear in the content descriptions.

This is an academic video-understanding benchmark task. The following is the benchmark question to answer:

Question:

> *[question]*
>
> Please read the provided video content descriptions and the question in depth, and determine whether you can accurately answer the question solely based on the currently provided descriptions.
>
> If you can answer it with absolute confidence, answer the question and provide the time periods to which you are referring. Your answer must have complete and objective support in the video descriptions. Do not make arbitrary inferences.
>
> If you think the current video descriptions are still insufficient to answer the question accurately, do not answer it and instead provide your reason.
>
> Please output a strictly standardized dictionary containing exactly four key–value pairs:
>
> - `"Confidence"`: A Boolean value. Set it to `True` if you are certain about the answer and `False` otherwise.
> - `"Answer"`: A string enclosed in double quotation marks. When `"Confidence"` is `True`, provide the correct answer. When `"Confidence"` is `False`, return `"No Answer"`. The answer must contain exactly one number: either the number of the selected option, if options are provided, or the numeric answer to the question, if no options are provided.
> - `"Time Period"`: A list. When `"Confidence"` is `True`, fill the list with the time periods corresponding to the answer, with each time period represented as a tuple in the format `(start time, end time)`. When `"Confidence"` is `False`, return `"No Time"`.
> - `"Reason"`: A string enclosed in double quotation marks. Explain the reasoning behind your judgment. Ensure that the reasoning provides complete and objective support for your answer.
>
> Do not add any additional comments within the dictionary.
>
> If an ordinal number appears in the provided question, you must note that, in the vast majority of cases, you should not simply assume that this ordinal number represents the ordinal position of a provided time period. Instead, focus on understanding the specific meaning of the ordinal number within the question based on all of the content descriptions.
>
> You **must** output a valid JSON dictionary.
>
> Return **only** a valid JSON object.
>
> The response must start with `{` and end with `}`.
>
> Do not wrap the JSON in Markdown fences.
>
> Do not add explanations before or after the JSON.
>
> Use double quotation marks for all keys and string values.
>
> Use JSON Boolean values `true`/`false`, not Python Boolean values `True`/`False`.
>
> For `"Time Period"`, use JSON arrays such as `[[120, 150]]`, not tuples such as `[(120, 150)]` or strings such as `"120s-150s"`.

We preserve VideoLucy's original localization-and-instruction prompt while making its existing exclusion of previously searched periods more explicit after having observed deviations from intended behavior on Minerva. Furthermore, we embed answer choices in the question (for cross-dataset evaluation), and enforce valid JSON outputs for reliable Gemini API inference.

Captioning Prompt

> You should explicitly and in detail describe any visible subtitles, text overlays, or voice-overs in the video. In addition, follow the instruction below: *[instruction]*

The intermediate captioning prompt is exactly identical to the one in the original repository.

Answering Prompt

There is currently a video with a total duration of *[video length]* seconds.

The following provides a general description of what is shown in the video during certain time periods:

Video descriptions:

[memory]

This is an academic video-understanding benchmark task. The following is the benchmark question to answer:

Question:

[question]

Please read the provided video content descriptions and the question in depth, and determine whether you can accurately answer the question solely based on the currently provided descriptions.

If you can answer it with absolute confidence, answer the question and provide the time periods of the video content to which you are referring. Your answer must have complete and objective support in the video descriptions. Do not make arbitrary inferences.

If you think the current video descriptions are still insufficient to answer the question accurately, do not answer it and instead provide your reason.

Please output a strictly standardized dictionary containing exactly four key–value pairs:

- `"Confidence"`: A Boolean value. Set it to `True` if you are certain about the answer and `False` otherwise.
- `"Answer"`: A string enclosed in double quotation marks. When `"Confidence"` is `True`, provide the correct answer option. When `"Confidence"` is `False`, return `"No Answer"`. The answer must contain exactly one number: either the number of the selected option, if options are provided, or the numeric answer to the question, if no options are provided.
- `"Time Period"`: A list. When `"Confidence"` is `True`, fill the list with the time periods corresponding to the answer, with each time period represented as a tuple in the format `(start time, end time)`. When `"Confidence"` is `False`, return `"No Time"`.
- `"Reason"`: A string enclosed in double quotation marks. Explain the reasoning behind your judgment. Ensure that the reasoning provides complete and objective support for your answer.

Do not add any additional comments within the dictionary.

If an ordinal number appears in the provided question, note that, in the vast majority of cases, you should not simply assume that this ordinal number represents the ordinal position of a provided time period. Instead, focus on understanding the specific meaning of the ordinal number within the question based on all of the content descriptions.

Return **only** a valid JSON object.

The response must start with `{` and end with `}`.

Do not wrap the JSON in Markdown fences.

Do not add explanations before or after the JSON.

Use double quotation marks for all keys and string values.

Use JSON Boolean values `true`/`false`, not Python Boolean values `True`/`False`.

For `"Time Period"`, use JSON arrays such as `[[120, 150]]`, not tuples such as `[(120, 150)]` or strings such as `"120s-150s"`.

We preserve VideoLucy's original answering policy while framing each query as an academic benchmark task to mitigate false-positive Gemini API filtering observed on Minerva. We further standardize predictions as dataset-agnostic numeric outputs and require valid JSON to ensure reliable parsing across datasets.

Forced Answer Prompt

> There is currently a video with a total duration of *[video length]* seconds.
>
> The following provides a general description of what is shown in the video during certain time periods:
>
> **Video descriptions:**
>
> *[memory]*
>
> This is an academic video-understanding benchmark task. The following is the benchmark question to answer:
>
> **Question:**
>
> *[question]*
>
> Please read and understand the provided video content and question in depth. Strictly based on the video content, select the single best answer option. You must provide a best answer for this question.
>
> If an ordinal number appears in the provided question, note that, in most cases, its meaning is unrelated to the ordinal position of a provided time period. Focus on analyzing the meaning of the ordinal number within the question.
>
> Please output a strictly standardized dictionary containing exactly three key–value pairs:
>
> - `"Answer"`: A string enclosed in double quotation marks. This must contain exactly one number. If answer options are provided, return the number of the selected option. If no answer options are provided, return the numeric answer to the question.
> - `"Time Period"`: A list containing the time periods corresponding to the best answer, with each period represented as a tuple in the format `(start time, end time)`.
> - `"Reason"`: A string enclosed in double quotation marks. Explain the reasoning behind your judgment. Ensure that the reasoning provides complete and objective support for your answer.
>
> Do not add any additional comments within the dictionary.
>
> Return **only** a valid JSON object.
>
> The response must start with `{` and end with `}`.
>
> Do not wrap the JSON in Markdown fences.
>
> Do not add explanations before or after the JSON.
>
> Use double quotation marks for all keys and string values.
>
> Use JSON Boolean values `true`/`false`, not Python Boolean values `True`/`False`.
>
> For `"Time Period"`, use JSON arrays such as `[[120, 150]]`, not tuples such as `[(120, 150)]` or strings such as `"120s-150s"`.

We preserve VideoLucy's forced-answer policy while adapting it for cross-dataset evaluation: questions are framed as academic benchmark tasks, answer choices are embedded directly in the question, predictions are standardized as numeric outputs, and responses are constrained to valid JSON for reliable Gemini API parsing.

A.3.3 VIDEO CURIOUS AGENT

The code for VCA is not public as of August 2026. We reimplemented the method as described in the paper. For prompts we follow the templates from the appendix of Yang et al. (2025c), however we encounter a few inconsistencies in the paper itself as detailed in the exploration prompt below:

Reward Model Prompt for first round

> You are acting as a reward model to guide the video question-answering process, with access to a `{self.total_frames}`-frame video

({self.video_duration_sec} seconds). You are provided with N uniformly sampled frames from the video, at the following frame indices: {",".join(map(str, frame_indices))}, which divide the video into $N + 1$ distinct segments.

Transcript aligned with these sampled frames:

{transcript}

{segment_text}

Your task is to evaluate the relevance of each segment in answering the question below, to assist in identifying the segment(s) that most effectively answer the question.

{question}

First provide an explanation based on the specific details observed in each sub-segment, then assign a relevance score from 0 to 100. Focus on each segment's alignment with the question.

Hint: Since only the start and end frames of each segment are provided, first imagine the possible content within each segment based on these frames before making a decision.

In your explanation, describe your reasoning process and any inferred details about the segment's content, using specific observations from the start and end frames as a basis for these inferences. Please give the answer in the format:

{"Segment #": {explanation: str, score: int}}

It needs to be a valid Python literal.

We preserve VCA's original tree-search, reward-guided exploration, and memory-management mechanisms. To ensure a controlled comparison, we additionally provide timestamp-aligned transcripts to VCA, as done for all evaluated methods. Prompts retain the original instructions, but explicitly require a standardized Python-literal output format to address inconsistencies observed in preliminary Minerva evaluations.

Reward Model Prompt for subsequent rounds

You are acting as a reward model in a multi-round video question-answering process. You have access to a {self.total_frames}-frame video ({self.video_duration_sec} seconds), along with results from a previous round of evaluation. In this round, one specific segment has been further divided to provide more detailed analysis. You are provided with N new sampled frames to assess these sub-segments in relation to the question, at the following frame indices:

{",".join(map(str, frame_indices))}

Question:

{question}

In the last round, the video was divided into multiple segments, and each segment was evaluated for its relevance to the goal question. Here are the results from all previous rounds:

{reward_history_text}

In this round, segment [{frame_indices[0]}, {frame_indices[-1]}] has been further explored with new uniformly sampled frames, dividing it into new sub-segments:

{segment_text}

Your task is to evaluate these new sub-segments for relevance to the original goal question based on provided frames, to assist in identifying the segment(s) that most effectively answer the question, while considering the context and results from the previous rounds.

First provide an explanation based on the specific details observed in each sub-segment, then assign a relevance score from 0 to 100. Focus on each segment's

alignment with the question, ensuring consistency with the scores and explanations from previous segments.

Transcript aligned with these sampled frames:

`{transcript}`

Hint: Since only the start and end frames of each segment are provided, first imagine the possible content within each segment based on these frames before making a decision.

In your explanation, describe your reasoning process and any inferred details about the segment's content, using specific observations from the start and end frames as a basis for these inferences. Please give the answer in the format:

```
{Segment #: {explanation: str, score: int}}
```

It needs to be a valid Python literal.

Again our prompt is almost identical to the original, we only require a stricter output format for consistency and add the transcript for a controlled evaluation.

Exploration Agent Prompt

You are a helpful assistant with access to a video that is `{self.total_frames}` frames long (`{self.video_duration_sec}` seconds).

`{question}`

You are tasked with exploring the video to gather the information needed to answer a specific question with complete confidence. At each step, you may select one segment of the video to examine. Once you choose a segment, you will receive a set of representative frames sampled from that segment. Use each exploration step strategically to uncover key details, progressively refining your understanding of the video's content. Continue exploring as needed until you have acquired all the information necessary to answer the question.

Transcript aligned with current memory frames:

`{transcript}`

Memory buffer frame indices:

`[{memory_frame_indices_text}]`

Candidate segments to choose from:

`{candidate_segments_text}`

For each candidate segment, an auxiliary video assistant has already evaluated the relevance score with respect to the question. Focus on the most promising segment if more exploration is needed.

If you need more exploration, respond with a valid Python literal in this format:

```
{"action": "explore", "start_idx": int, "end_idx": int}
```

If you already have enough information, respond with a valid Python literal in this format:

```
{"action": "answer", "answer": int, "reasoning": str}
```

for multiple choice questions, where the answer is the index of the correct choice.

```
{"action": "answer", "answer": float, "reasoning": str}
```

for questions requiring a numerical answer.

Hint: Do not rush to provide an answer. Take time to verify details and gather sufficient information before concluding. Approach each question as a step toward building a comprehensive understanding, ensuring accuracy over speed.

You must always return a valid Python literal.

This prompt is based on Figure 8 of the original paper Yang et al. (2025c). We identify an ambiguity between the presented prompt and the algorithmic description: Figure 8 displays only the subsegments created during the current round, whereas Section 3.3 and Algorithm 1 specify that the exploration agent receives the accumulated candidate set (S), comprising both newly created fine-grained segments and unexplored segments from previous rounds. The latter interpretation is necessary to support the backtracking behavior described in the paper. We therefore follow Algorithm 1 and provide the accumulated candidate set. The remaining instructions follow the original template, apart from the addition of timestamp-aligned transcripts and a standardized output format for reliable parsing.

A.3.4 VIDEOAGENT

For the VideoAgent implementation we use chat functionality from Gemini (except for captioning), where for each question the model retains access to all prior messages sent during the question answering process.

Answering Prompt (Initial Prompt)

> You will be given a question about a video and some possible answer options. You are provided descriptions of frames sampled from the video at relevant points. Frame Descriptions: `{captions_text}` Transcript: `{transcript}` Question: `{question}` After explaining your reasoning, output the final answer in the format "Final Answer: (X)" where X is the correct digit choice. Never say "unknown" or "unsure", or "None", instead provide your most likely guess.

The caption prompt from the original repository [4] includes EgoSchema-specific hints such as "#C" for the camera wearer and #O for someone other than the camera wearer. For the cross-dataset evaluation we remove those hints, while keeping the semantics of the prompt intact. In addition, as with all other baselines, we add the ASR-generated transcript to the prompt. We also adjust the output format to our evaluation. Preliminary Minerva runs revealed that VideoAgent tends to return "unknown" frequently; to ensure a fair evaluation with other baselines (which are forced to provide a guess), we prohibit VideoAgent from returning an invalid answer .

Self-Evaluation Prompt

> Please assess the confidence level in your decision-making process for the answer you just provided: "`{answer}`" Criteria for Evaluation: Insufficient Information (Confidence Level: 1): If information is too lacking for a reasonable conclusion. Partial Information (Confidence Level: 2): If information partially supports an informed guess. Sufficient Information (Confidence Level: 3): If information fully supports a well-informed decision. Assessment Focus: Evaluate based on the relevance, completeness, and clarity of the provided information in relation to the decision-making context. Return only valid JSON in the format `{confidence_format}`. The confidence value must be the integer 1, 2, or 3, not a label or explanation.

Apart from a stricter output format, this prompt is identical to the original one from the repository [5].

Captioning Prompt

> You are an expert video analyst. Describe this video frame in detail, specifically focusing on information relevant to the following context:
> `{condition}`
> In your description, focus on:
> - Main subjects/objects visible
> - Actions or activities occurring

[4] https://github.com/wxh1996/VideoAgent
[5] https://github.com/wxh1996/VideoAgent

- Scene/environment description
- Any text visible
- Relevant spatial relationships

Provide a concise but comprehensive description in 2-3 sentences.

The original repository and paper Wang et al. (2024) do not provide the prompt used to generate captions; instead, they release only precomputed captions for EgoSchema. We therefore introduce the task-conditioned captioning prompt above to generate frame descriptions in our implementation.

Segment Generation Prompt

So the information in our previous frames was not sufficient. **Objective:** Our goal is to identify additional frames that contain crucial information necessary for answering the question. These frames should not only address the query directly but should also complement the insights from the previous frames. To achieve this, we will: 1. Divide the video into candidate segments based on the intervals between the initial frames. Candidate Segments (ID: Start-End in Indices): `{segment_des}` 2. Determine which segments are likely to contain frames that are most relevant to the question. These frames should capture key visual elements (objects, humans, interactions, actions, scenes) that help answer the question. **Instructions:**

- For each frame identified as potentially relevant, provide a concise description focusing on essential visual elements.
- Use a single sentence per frame.
- If the specifics of a segment's visual content are uncertain based on the current information, use placeholders for specific actions or objects, but ensure the description still conveys the segment's relevance to the query.
- You may select multiple frames from one segment if necessary.

Return the descriptions and the `segment_id` in JSON format.

- `"segment_id"` must be an integer from the Candidate Segments keys (1 to `{len(segment_des)}`).
- `"duration"` must match the duration string from the chosen candidate segment.

Example Output:

```
{ "frame_descriptions": [ { "segment_id": 1, "duration": "xxx - xxx", "descrip
```

Output just the JSON object, nothing else.

Semantically, this prompt is equivalent to the original. The caption and question are retained from previous prompts through the chat functionality.

A.3.5 GEMINI NATIVE

Multiple-choice prompt

You will be given a question about a video and five possible answer options. Watch the video carefully to answer the question. Question: `{question}` After explaining your reasoning, output the final answer in the format "Final Answer: (X)" where X is the correct digit option choice. Never say "unknown" or "unsure", or "None", instead provide your most likely guess.

Numeric questions prompt (VSIBench)

You are an expert at spatial reasoning. Watch the video carefully and answer the following numerical question. Think step by step before giving your final answer. `{question}` Output your final answer as a number in the format "Final Answer: <number>" (e.g., "Final Answer: 42.5").

Both prompts are adapted from the original Minerva paper Nagrani et al. (2025).

A.3.6 DEEPVIDEODISCOVERY

To adapt DeepVideoDiscovery to the Gemini API we use the Gemini analogue of OpenAI tools, `types.Tool`.

Captioning Prompt

> There are consecutive frames from a video. Please understand the video clip with the given transcript then output JSON in the template below.
>
> **Transcript of current clip:**
>
> `TRANSCRIPT_PLACEHOLDER`
>
> **Output template:**

```
{
  "clip_start_time": "CLIP_START_TIME",
  "clip_end_time": "CLIP_END_TIME",
  "subject_registry": {
    <subject_i>: {
      "name": <fill with short identity if name is unknown>,
      "appearance": <list of appearance descriptions>,
      "identity": <list of identity descriptions>,
      "first_seen": <timestamp>
    },
    ...
  },
  "clip_description": <smooth and detailed natural narration of the video clip>
}
```

> Remember your output MUST be a valid JSON.

We use a near-identical copy of the prompt from the original repository [6], with only two formatting-related modifications: timestamp placeholders are quoted, and the model is explicitly instructed to produce valid JSON. We observed that this significantly increased output reliability, necessary for parsing model responses.

Subject-registry merge prompt

> You are given several partial `new_subject_registry` JSON objects extracted from different clips of the *same* video. They may contain duplicated subjects with slightly different IDs or descriptions.
>
> **Task:**
>
> 1. Merge these partial registries into one coherent `subject_registry`.
> 2. Preserve all unique subjects.
> 3. If two subjects obviously refer to the same person, merge them (keep earliest `first_seen` time and union all fields).
>
> The merge prompt is an exact copy of the original prompt.
>
> **Main Agent Prompt**
>
> > You are a helpful assistant who answers multi-step questions by sequentially invoking the provided function tools. Follow the THINK → ACT → OBSERVE loop:
> >
> > - **THOUGHT** Reason step-by-step about which function to call next.
> > - **ACTION** Invoke exactly one provided function tool that moves you closer to the final answer.
> > - **OBSERVATION** Summarize the function's output.

[6] `https://github.com/microsoft/DeepVideoDiscovery/tree/64414b2f35d26809a39740a5a319889f46e29b94`

Use native function calling only. Do not write code, `print(...)`, `default_api` calls, or textual ACTION calls.
You MUST plan extensively before each function call, and reflect extensively on the outcomes of the previous function calls.
Only pass arguments that come verbatim from the user or from earlier function outputs—never invent them. Continue the loop until the user's query is fully resolved, then end your turn with the final answer. If you are uncertain about code structure or video content, use the available tools to inspect rather than guessing. Plan carefully before each call and reflect on every result. Do not rely solely on blind function calls, as this degrades reasoning quality. Timestamps may be formatted as `'HH:MM:SS'` or `'MM:SS'`.
Carefully read the timestamps and narration in the following script, paying attention to the causal order of events, object details and movements, and people's actions and poses.
Here are tools you can use to reveal your reasoning process whenever the provided information is insufficient.

- To get a global information about events and main subjects in the video, use `global_browse_tool`.
- To search without a specific timestamp, use `clip_search_tool`.
- If the retrieved material lacks precise, question-relevant detail (e.g., an unknown name), call `frame_inspect_tool` with a list of time ranges (`list[tuple[HH:MM:SS, HH:MM:SS]]`).
- Whenever you are uncertain of an answer after searching, inspect frames in the relevant intervals with `frame_inspect_tool`.
- After locating an answer in the script, always make a **CONFIRM** with `frame_inspect_tool` query.

You can first use `global_browse_tool` to a global information about this video, then invoke multiple times of these tools to progressively find the answer.
Based on your observations and tool outputs, provide a concise answer that directly addresses the question.
ALWAYS invoke one of the provided function tools. Do not write tool calls as text. If you're done, invoke the `'finish'` tool.
After either you've called the `'finish'` tool or you've reached the max number of iterations, you'll be prompted to answer the question. Here please DON'T call a tool just provide your answer in text.
Total video length: `VIDEO_LENGTH` seconds.
Question: `QUESTION_PLACEHOLDER`

]
Input (list of JSON objects):
`REGISTRIES_PLACEHOLDER`
Return *only* the merged `subject_registry` JSON object.

We adjust the wording slightly to use Gemini terminology (`tools` instead of `functions`). Furthermore, we encountered several cases in which the model produced textual tool calls or answered prematurely instead of using Gemini's native function-calling interface. We therefore explicitly require native tool calls and instruct the model to invoke the `finish` tool before producing its final answer. Apart from these interface-specific adjustments, the original reasoning and tool-use instructions remain unchanged.

Global-browse Prompt

You are a knowledgeable assistant specializing in analyzing video content and providing detailed, insightful answers.
Below are descriptions of video clips, each with its corresponding timestamp. Carefully review the sequence of events, the details and movements of objects,

> and the actions and poses of people. Based on these observations, provide a thorough and specific answer to the following question, referencing key events and timestamps as appropriate.
>
> Question: `{query}`
>
> `{clip_captions}`

No changes were made to this prompt.

Frame-inspection Prompt

> Carefully watch the video frames. Pay attention to the cause and sequence of events, the detail and movement of objects and the action and pose of persons.
>
> Based on your observations, if you find content that can answer the question. If no relevant content is found within the given time range, return:
>
> `Error: Cannot find corresponding result in the given time range.`
>
> Question: `{question}`

Again, we do not modify the original prompt.

A.3.7 A.I.R.

We copy the algorithm from the authors' official GitHub [7] repository adapting only the backbone calls as required by our evaluation protocol.

Analysis Prompt

> You are an expert visual reasoner. Step by step analyze how well the provided images match the following query (with options):
>
> Query: `{query}`
>
> Rate the relevance from 1 to 5, using these exact definitions:
>
> 1 – Not relevant at all: no relation between image content and the query.
>
> 2 – Slightly relevant: only minor contextual hints, but not central to answering.
>
> 3 – Moderately relevant: contains preparatory or follow-up context (e.g. setup actions) related to the query.
>
> 4 – Highly relevant: shows clear evidence to support query but still has ambiguity or missing detail.
>
> 5 – Perfectly matches: fully sufficient and unambiguous evidence to answer the query.
>
> Avoid overthinking. Trust your immediate judgment.
>
> Examples:
>
> – Image of an empty room (no people or objects) → Reasoning: There is nothing related to the query. Score: 1
>
> – Image showing exactly the queried action (clear, direct match) → Reasoning: It perfectly depicts the required event. Score: 5
>
> Respond with exactly one valid JSON array and nothing else.
>
> Requirements:
>
> - Return exactly one object for every provided frame.
> - Use the exact zero-based Index shown in each frame header.
> - Score must be an integer from 1 through 5.
> - Do not include text before or after the JSON array.
>
> Example:

[7] https://github.com/UCF-AIR/A.I.R./tree/main

```
[
    {
        "Index": 0,
        "Score": 4,
        "Reasoning": "The frame clearly shows evidence relevant to the questio
    }
]
```

The analysis prompt is copied verbatim from the repository, only the output formatting is adjusted to ensure robustness to common parsing and consistency failures encountered during evaluation. We specifically require the model to output its analysis in JSON format, consistent with requirements for other baselines.

B Application of the Active Exploration Formalism to Existing Approaches

B.1 Active Exploration Baseline

We subsequently present one of the simplest formalizations of the agentic video reasoning framework which we use to construct the Active Exploration Baseline.

Action Space and Observations The action space consists of the simplest conceivable operator:

$$\mathcal{A}_{\text{query}} = \{\textsc{Query}(n)\}$$

Query takes in a frame index n and simply returns the corresponding frame from the video. The observation operator O therefore returns the frame at index n:

$$o_{t+1} = O(\textsc{Query}(n), E) = x_n$$

where x_n is the frame at index n.

Initialization We don't use extensive video preprocessing, F_{init} outputs the first frame index to inspect, along with the agent's reasoning r_1 (without having seen any frames):

$$s_1 = F_{\text{init}}(q, E) = (n_1, r_1) \qquad \pi(\textsc{Query}(n_1) \mid s_1) = 1.$$

State Update The state update is the primary exploration step. The agent chooses the next action and outputs its internal reasoning, both of which are concatenated with all prior reasoning outputs, giving the interaction history up to iteration $t+1$, r_{t+1}.

$$s_{t+1} = F(q, s_t, o_{t+1}) = (a_{t+1}, r_{t+1})$$

a_{t+1} is the next action, either Stop or $\textsc{Query}(n_{t+1})$.

Policy and termination The policy then deterministically extracts a_{t+1} from s_{t+1}. Upon termination the agent executes a final reasoning step based on the interaction history and outputs a final answer. τ denotes termination time:

$$\hat{y} = g(s_\tau) \in \mathcal{Y}.$$

B.2 VideoAgent

Action Space and Observations In VideoAgent Wang et al. (2024) the actions are presented as:

$$\mathcal{A}_{\text{query}} = \{\textsc{UniformSampling}, \textsc{SegmentSearch}(Q)\}$$

Here, UNIFORMSAMPLING captions uniformly sampled frames from the full video, while SEGMENTSEARCH takes in a set of queries $Q = (q_i, seg_i)$. For each query (q_i, seg_i), where q_i denotes a textual query and seg_i specifies a temporal segment in the video, the action retrieves the frame within the corresponding segment, which is most similar according to CLIP Radford et al. (2021) similarity. The observation operator O returns the requested captions.

Initialization Since VideoAgent does not include a dedicated preprocessing step, we model initialization as returning a fixed start state. Concretely, we set

$$s_1 = F_{\text{init}}(q, E) = s^{(0)}$$

where $s^{(0)}$ is a special START token such that the policy selects UNIFORMSAMPLING with probability one:

$$\pi(\text{UNIFORMSAMPLING} \mid s^{(0)}) = 1$$

State Update The state update F is VideoAgent's internal reasoning step: At time t it tries to answer the question and assess its own confidence. If the confidence surpasses a certain threshold STOP is invoked and the final answer is returned in s_t, otherwise the process continues with SEGMENTSEARCH. In either case the new captions are added to the internal memory from $s_{\text{t-1}}$. s_t consists of the frame captions, the answer to the user's question and the agent's confidence in the answer.

VideoAgent's policy is deterministic:

$$\text{Either} \qquad \pi(\text{STOP} \mid s_t) = 1 \quad \text{or} \quad \pi(\text{SEGMENTSEARCH} \mid s_t) = 1.$$

depending on which action the agent chose in F. Note that VideoAgent uses UNIFORMSAMPLING only for initialization; our framework also permits a straightforward extension where UNIFORMSAMPLING becomes a reusable query action.

Final prediction The prediction head just extracts the correct answer from the final agent state s_τ

$$\hat{y} = g(s_\tau)$$

B.3 VIDEO CURIOUS AGENT

VCA Yang et al. (2025c) treats a long video as an environment to explore via a tree-search over temporal segments. At each iteration it uniformly samples frames from the currently selected segment, scores the resulting subsegments using an intrinsic reward model, updates a fixed-size memory buffer, and then either outputs an answer or selects a new segment for further exploration.

Action space. In our formalism, VCA's query actions correspond to selecting the next temporal segment to explore. Let a segment be parameterized by its temporal bounds (ℓ, r). We write

$$\mathcal{A}_{\text{query}} = \{\text{SELECTSEGMENT}(\ell, r)\}, \qquad \mathcal{A} = \mathcal{A}_{\text{query}} \cup \{\text{STOP}\}.$$

Observation operator. Given the current segment (ℓ, r), VCA observes the environment by sampling N frames uniformly from that segment Yang et al. (2025c). We model this as

$$o_{t+1} = O(\text{SELECTSEGMENT}(\ell_t, r_t), E) = \text{UNIFORMSAMPLE}(\ell_t, r_t),$$

where $o_{t+1} \in \mathcal{O}$ contains the sampled frames.

Initialization. VCA initializes the agent state to contain uniformly sampled frames across the whole video. We capture this by an initializer

$$s_1 = F_{\text{init}}(q, E) = s^{(0)},$$

where $s^{(0)}$ is a special START token such that the policy selects SEGMENTSEARCH on the whole video with probability one:

$$\pi(\textsc{UniformSampling}, \ell_0 = 0, r_0 = N \mid s^{(0)}) = 1$$

where N is the last frame of the video.

State update. VCA's state update corresponds to (i) computing intrinsic reward scores for candidate subsegments using a reward model R, (ii) updating the candidate set and reward-history (lines 12–14), and (iii) updating a fixed-size memory buffer with sampled frames (line 16) Yang et al. (2025c). In our notation,

$$s_{t+1} = F(q, s_t, o_{t+1}),$$

where s_t can be understood as the agent's state, containing candidate segments, the reward history, the fixed-size memory buffer, and the agent's answer to the question.

Policy and termination. The induced policy selects either STOP or $\textsc{SelectSegment}(\cdot)$; termination occurs when a non-empty answer is present in s_{t+1}.

Prediction head. Upon termination at time τ, the agent has returned a non-empty answer which is extracted from s_τ:

$$\hat{y} = g(s_\tau).$$

B.4 DEEP VIDEO DISCOVERY (DVD)

Deep Video Discovery (DVD) Zhang et al. (2025b) frames long-video understanding as iterative information gathering over a multi-granular video database using a search-centric toolset and an LLM orchestrator. Concretely, the agent iteratively selects an action from a toolset (or answers) and updates its internal reasoning based on the returned tool outputs.

Action space DVD exposes three query tools—GLOBALBROWSE, CLIPSEARCH, and FRAMEINSPECT, in addition to the STOP action from our formalism. We therefore model

$$\mathcal{A}_{\text{query}} = \{\textsc{GlobalBrowse}(s_t, q),\ \textsc{ClipSearch}(s_t, \hat{q}),\ \textsc{FrameInspect}(s_t, \hat{q}, I)\}$$
$$\mathcal{A} = \mathcal{A}_{\text{query}} \cup \{\textsc{Stop}\}.$$

Here $s_t \in \mathcal{S}$ denotes the current agent state (memory / context), $q \in \mathcal{Q}$ is the original user query, and $\hat{q}$ is a refined sub-query produced by the agent (e.g., a more specific retrieval prompt). Finally, I denotes a temporal interval to inspect, e.g. $I = [t_s, t_e]$. Intuitively, GLOBALBROWSE produces a coarse global summary conditioned on (s_t, q), CLIPSEARCH retrieves relevant clips for $\hat{q}$, and FRAMEINSPECT performs a targeted inspection within I for $\hat{q}$.

Observation operator The observation operator O returns the tool outputs: (i) GLOBALBROWSE returns global summaries (subject-centric and event-centric), (ii) CLIPSEARCH returns top-k relevant clips along with captions and time ranges via caption-embedding retrieval, and (iii) FRAMEINSPECT returns an open-format VQA response for a specified time window and sub-query. Formally, for a tool action $a_t \in \mathcal{A}_{\text{query}}$,

$$o_{t+1} = O(a_t, E) \in \mathcal{O}.$$

Initialization For DVD, the agent first constructs a multi-granular video database that is subsequently queried by the tool actions.

$$s_1 = F_{\text{init}}(q, E).$$

Importantly, this preprocessing step is query-agnostic, so F_{init} discards the input q. The resulting state s_1 stores the multi-granular video representations that are used by the subsequent tool calls.

State update The state update F corresponds to the LLM's iterative reasoning and memory update: it integrates the new observation o_{t+1} (tool output) into the current reasoning context and decides what information is still missing:

$$s_{t+1} = F(q, s_t, o_{t+1}) \in \mathcal{S}.$$

The updated state s_{t+1} contains the database as well as the interaction history, consisting of reasoning traces, actions, and observations. Consequently, the size of s_{t+1} grows linearly with the number of interaction steps.

Policy and termination DVD's policy π is deterministic and directly extracts the action from the previous state.

Final prediction Upon termination at time τ, the final answer is extracted from the terminal state s_τ.

B.5 VIDEOTREE

VideoTree Wang et al. (2025c) constructs a query-adaptive hierarchical representation of the video by first building a set of query-relevant top-level clusters and then expanding only the relevant clusters to finer levels before performing final LLM reasoning.

Action space In our formalism, VideoTree's query actions correspond to operations that iteratively refine a tree-structured representation of the video. Instead of directly selecting frames or segments, the agent acts by modifying this tree.

We introduce four key variables: k, the number of clusters used at the current breadth expansion step; C, a cluster (i.e., a node in the tree); w, the branching factor; and d, the depth of expansion. Based on this, we define the query action space as follows:

$$\mathcal{A}_{\text{query}} = \{\textsc{BreadthExpand}(k),\ \textsc{DepthExpand}(C, w, d),\ \textsc{CaptionKeyframes}\},$$
$$\mathcal{A} = \mathcal{A}_{\text{query}} \cup \{\textsc{Stop}\}.$$

Here, BREADTHEXPAND(k) performs visual clustering of the video into k clusters and returns the caption of each keyframe; DEPTHEXPAND(C,w,d) refines a selected cluster C into a subtree of width w and depth d; and CAPTIONKEYFRAMES traverses the constructed tree, extracts representative keyframes from its clusters, and returns their captions in temporal order.

Observation operator The observation operator returns the outputs of these refinement steps:

$$o_{t+1} = O(a_t, E) \in \mathcal{O}.$$

Concretely: (i) BREADTHEXPAND(k) returns top-level clusters together with representative captions; (ii) DEPTHEXPAND(C,w,d) returns a refined subtree of depth d rooted at C, obtained by iteratively re-clustering the frames in C; and (iii) CAPTIONKEYFRAMES returns captions for the keyframes in temporal order.

Thus, $\mathcal{O}$ contains structured observations such as clusters, captions and subtrees, rather than only raw frames.

Initialization VideoTree initializes the agent state with a fixed start token. Concretely, we set

$$s_1 = F_{\text{init}}(q, E) = s^{(0)},$$

where $s^{(0)}$ is a special START token such that the policy selects BREADTHEXPAND for the initial clustering with probability one:

$$\pi(\textsc{BreadthExpand}(k_0) \mid s^{(0)}) = 1.$$

State update The state update

$$s_{t+1} = F(q, s_t, o_{t+1})$$

corresponds to updating the current tree representation and its associated metadata. The state s_t contains the partially constructed VideoTree, including current clusters, their relevance scores, representative keyframes and captions. During adaptive breadth expansion, F updates the first-level clustering by providing the relevance scores between 1 and 3 for each cluster. If the observation o_{t+1} contains the output of CAPTIONKEYFRAMES, the state update F stores these captions in a temporally ordered reasoning context within s_{t+1} and prepares the agent for termination.

Policy and termination VideoTree's induced policy is deterministic. If the state update F indicates that the number of highly relevant clusters (relevance score 3) is below a threshold and k is below the maximum breadth, the policy chooses BREADTHEXPAND(k). Otherwise, it selects DEPTHEXPAND(C,w,d) for clusters based on the relevance scores assigned during the state update: $d = 1$ for score 2 and $d = 2$ for score 3. Once the tree refinement is complete, the policy selects CAPTIONKEYFRAMES; upon receiving the final captions, it chooses STOP.

Final prediction Upon termination at time τ, the final answer is produced from the terminal state:

$$\hat{y} = g(s_\tau).$$

Here, g corresponds to the LLM reasoning module that consumes the temporally ordered captions extracted from the constructed tree together with the original query and outputs the answer.

B.6 VIDEOLUCY

VideoLucy Zuo et al. (2025) addresses long-video question answering through a deep backtracking mechanism. It builds a coarse video memory by captioning the full video at a low temporal resolution, and then iteratively re-examines the most relevant temporal periods at progressively finer granularity until a confident answer emerges.

Action space At each step, VideoLucy selects a temporal period to re-examine together with a targeted captioning instruction. We define

$$\mathcal{A}_{\text{query}} = \{\text{RECAPTIONPERIOD}(p, I) \mid p = (s, e) \subseteq [0, T],\ I \in \mathcal{I}\},$$
$$\mathcal{A} = \mathcal{A}_{\text{query}} \cup \{\text{STOP}\}.$$

Here $p = (s, e)$ is a temporal interval and I is a natural-language captioning instruction specifying which visual details to focus on.

Observation operator Given action RECAPTIONPERIOD(p, I), the observation operator produces hierarchical captions of period p:

$$o_{t+1} = O(\text{RECAPTIONPERIOD}(p, I), E) \in \mathcal{O}.$$

Concretely, o_{t+1} consists of (i) an *entire* caption of the full period p at a resolution determined by its duration, and (ii) *divided* captions of non-overlapping sub-clips of p at a finer temporal resolution.

Initialization VideoLucy's initialization is a two-phase preprocessing step.

Specifically $\mathrm{CM} = \{(p_k, d_k)\}_{k=1}^{K_c}$ is first constructed where each entry stores a temporal period p_k and its caption d_k. In F_{init}, the agent tries to answer the question based solely on CM. Two cases are distinguished: (1) If the agent can confidently answer the question just based on CM, F_{init} returns a special termination state $s^{(0)}$. (2) Otherwise, CM is internally evaluated against the query q to extract the top-K most relevant periods, forming a sparse coarse memory $\mathrm{CM}_{\text{init}} \subseteq \mathrm{CM}$ which populates the standard initial state s_1 as the working memory for the subsequent exploration loop. We can therefore write:

$$s_1 = F_{\text{init}}(q, E).$$

State update At each iteration, the state update F integrates the newly acquired fine-grained observation o_{t+1} into a three-level hierarchical memory. Then, an Answering Agent evaluates whether this updated memory suffices to answer the question confidently. Consequently, the new state

$$s_{t+1} = F(q, s_t, o_{t+1}).$$

stores the updated hierarchical memory (combining $\mathrm{CM}_{\text{init}}$ with fine and super-fine histories), the drafted answer, and the Answering Agent's confidence score.

Policy and termination If the initialization returns the special state $s^{(0)}$, the policy immediately terminates with $\pi(\textsc{Stop} \mid s^{(0)}) = 1$. Otherwise, for a working state s_{t+1}, the policy's distribution is determined by the recorded confidence. If the Answering Agent was confident, the system terminates. Otherwise, a Localization+Instruction Agent processes s_{t+1} to select the single most relevant unexplored period and produce a targeted captioning instruction, determining the next $\textsc{RecaptionPeriod}(p^*, I^*)$ action:

$$\pi(\textsc{Stop} \mid s_{t+1}) = 1 \quad \text{if } s_{t+1} \text{ is confident, else}$$
$$\pi(\textsc{RecaptionPeriod}(p^*, I^*) \mid s_{t+1}) = 1.$$

Final prediction Upon termination at time τ, the final answer is extracted from the terminal state:

$$\hat{y} = g(s_\tau).$$

C Validation Against Prior Baseline Results

To sanity-check our adaptations of the baselines to the new backbones, we compare our Gemini 3 Flash and Gemma 4 results to publicly reported results of the same baselines using other backbones.

VideoAgent: Our VideoAgent implementation with Gemini 3 Flash achieves **64.2%** accuracy on the 500-question EgoSchema subset. The original VideoAgent paper Wang et al. (2024) reports **60.2%** using GPT-4, while VCA reports **63.2%** with a GPT-4o-based reimplementation and **65.2%** using Qwen2-VL-72B. Gemini 3 Flash is a substantially newer and stronger multimodal model than the GPT-4 and GPT-4o backbones used in earlier evaluations, as indicated by recent multimodal benchmarks Yue et al. (2025). Similarly, our VideoAgent implementation with Gemma 4, which is a frontier open-weight model, achieves **67.7%** on EgoSchema. The modest improvements over the original GPT-4 result and the GPT-4o reimplementation are therefore directionally consistent with the use of newer backbones. At the same time, the differences are small: on the 500-question subset, one percentage point corresponds to only five examples. Furthermore, direct comparability to prior results is limited, as we are only able to evaluate on 491 of the original 500 EgoSchema samples, and the VCA reimplementation is not publicly available. We therefore interpret these results primarily as evidence that our implementation reproduces the expected VideoAgent performance regime.

VideoTree: Our VideoTree implementation achieves **73.1%** accuracy on the EgoSchema subset with Gemini 3 Flash and **71.1%** with Gemma 4. These results are consistent with previously reported VideoTree evaluations using different reasoning backbones. The original VideoTree paper Wang et al. (2025c) reports **66.2%** with GPT-4, while VCA Yang et al. (2025c) reports **67.0%** with a GPT-4o-based reimplementation and **71.0%** using Qwen2-VL-72B. Our Gemini 3 Flash and Gemma 4 results are therefore broadly consistent with prior VideoTree reimplementations, with the improvement over GPT-4/GPT-4o being directionally consistent with the use of newer and stronger multimodal backbones.

VideoLucy: We could not identify prior VideoLucy evaluations on EgoSchema, Minerva, or VSI-Bench. Consequently, no directly comparable published results are available for validating our VideoLucy reimplementation on these benchmarks.

VideoCuriousAgent (VCA): VCA does not provide an official public implementation, requiring us to reimplement the method from the paper description and prompts. Prior reported EgoSchema results include **73.6%** with GPT-4o and **75.2%** with Qwen2-VL-72B in the original VCA paper Yang et al. (2025c), as well as **74.1%** from the independent EEA reimplementation using Seed1.6VL Yang et al.

(2025b). Our implementation achieves **72.0%** with Gemini 3 Flash and **67.7%** with Gemma 4. Notably, Gemini 3 Flash and Seed1.6VL exhibit broadly comparable performance under the controlled SIS-Bench evaluation Zou et al. (2026b), further validating the consistency of our Gemini 3 Flash result with the independently reported Seed1.6VL reimplementation. Due to the prohibitive computational cost of unrestricted iterative exploration, we impose a maximum number of VCA iterations. Neither the original VCA paper nor the EEA reimplementation reports whether a hard iteration limit was used, making this aspect of their evaluation setup unclear. Moreover, we evaluate on only 491 of the original 500 EgoSchema samples. Consequently, the results are not directly comparable, but remain in the same general performance regime as prior VCA implementations.

DeepVideoDiscovery (DVD): The original DVD paper reports **76.6%** on EgoSchema using GPT-4.1-mini to construct the textual video database and o3 for agentic reasoning and frame inspection. Our implementation replaces these components with Gemini 2.5 Flash-Lite and Gemini 3 Flash, respectively, and achieves **69.1%**. This gap is consistent with the strong backbone sensitivity reported by DVD itself: on LVBench, replacing o3 with weaker reasoning models reduces performance by several percentage points. Since our adaptation changes both the database-construction and reasoning backbones, an exact reproduction of the original score is not expected. We therefore view the observed result as consistent with the performance regime reported for DVD under different model configurations.

Gemini 3 Flash: The Seed2.0 model card Seed (2026) reports Gemini 3 Flash native performance on Minerva at 64.4%, very close to the 64.8% obtained with our implementation. This close agreement with an independently reported evaluation provides additional validation of our evaluation pipeline.

D Full Results

A full breakdown of the results is shown in Table 3.

E SAVE PROMPTS AND ALGORITHM

E.1 ALGORITHM

Algorithm 1 SAVE: Simple Agentic Video Explorer

Require: Video $V = x_i{}_{i=1}^{N}$, question q, backbone M, query budget B, maximum returned scenes K, compression interval C, high-resolution threshold δ

$h \leftarrow [\text{video metadata}, q]$ ▷ Text-only exploration history
$(r_0, a_1) \leftarrow M(h)$ ▷ Initial reasoning and action
$h \leftarrow \textsc{Append}(h, r_0)$
for $t = 1, \ldots, B$ **do**
  if $a_t = \textsc{Stop}$ **then**
    break
  end if
  $(s, e) \leftarrow a_t$ ▷ $a_t = \textsc{QueryRange}(s, e)$
  $\mathcal{S} \leftarrow \textsc{SceneDetect}(V[s:e])$ ▷ Adapt the detection threshold to the queried range
  $\mathcal{S}' \leftarrow \textsc{UniformSubsample}(\mathcal{S}, K)$ ▷ Restrict number of scenes to K
  $I \leftarrow \{\text{mid}(S) \mid S \in \mathcal{S}'\}$
  $d_t \leftarrow |e - s| / \text{fps}(V)$
  $\rho_t \leftarrow \begin{cases} \textsc{High}, & d_t \le \delta, \\ \textsc{Medium}, & d_t > \delta \end{cases}$ ▷ Use higher visual fidelity for narrow ranges
  $F_t \leftarrow \textsc{LoadFrames}(V, I, \rho_t)$ ▷ Load selected frames and transcripts at resolution ρ_t
  $o_t \leftarrow \{(i, F_t[i]) \mid i \in I\}$
  $(r_t, a_{t+1}) \leftarrow M(q, h, o_t)$ ▷ Observe, reason, and select the next query range
  $h \leftarrow \textsc{Append}(h, r_t)$
  if $t \bmod C = 0$ **then**
    $h \leftarrow \textsc{Compress}(h, q)$ ▷ Retain relevant evidence and searched-range anchors
  end if
end for
$\hat{y} \leftarrow M.\textsc{FinalAnswer}(q, h)$
return $\hat{y}$

E.2 PROMPTS

Agent prompt. The active vision agent was prompted to iteratively gather visual evidence before answering. At each step, it received the video metadata, the question, and the following action interface:

> You are an Active Vision Agent.
>
> Answer the video question using as few visual queries as needed. Cost matters, but accuracy matters more; gather enough evidence before answering.
>
> Video: [number of frames] frames, FPS [fps], duration [duration]s. Frame indices are original frame numbers from 0 to [last frame index]; timestamp = index / [fps].
>
> Output exactly: **Observation**, **Thought**, then one **Action**.
>
> **Observation:** Describe only relevant facts from the new frames, with exact frame indices.
>
> **Thought:** Explain the next most useful evidence to gather or why you can answer.
>
> **Action:** choose exactly one:
>
> - `ACTION: QUERY <start_index> <end_index>`: inspect that frame range and return relevant visual observations plus transcript. Example: `ACTION: QUERY 0 3000`.
>
> A QUERY returns selected sampled frames from the requested range, not every frame. For broader queried ranges, fine details or brief events may be missed between sampled frames. If answer-changing details remain uncertain, query a narrower range.

- `ACTION: STOP`: stop only when you're certain. Do not STOP if two answers remain plausible from the visual evidence. If any answer-changing uncertainty remains, use `QUERY`.

Sometimes a precise answer might not be possible (e.g., numerical questions); in that case, always provide your best estimate and stop once you're reasonably confident. Never provide an empty or invalid answer.

Question: "[question]"

Do not hallucinate.

History compression prompt. Interaction history was compressed with a separate prompt that preserved navigation anchors while removing irrelevant visual detail:

You are an expert at information compression. Compress the interaction history below. The agent is answering the given question.

Rules:

1. Preserve every `[OBSERVATION: Indices ...]` header exactly; these are navigation anchors showing which video segments were searched.
2. For observations where the agent found nothing relevant, replace the body with one short phrase, e.g. "Irrelevant — <3-word reason>." No visual details.
3. For observations where the agent found something relevant, write 1–2 terse sentences capturing only the key facts that help answer the question. Drop all descriptions irrelevant to the question. Keep precise frame indices when relevant.
4. End with a single `Status:` line summarising the current hypothesis and what region to search next.
5. Output only the compressed result, with no preamble or meta-commentary.

F Further Analysis

F.1 Statistical Significance

On pairwise common error-free Minerva samples, two-sided video-cluster permutation tests found that SAVE outperformed VideoLucy, VideoTree, VideoCuriousAgent, VideoAgent, DeepVideoDiscovery, A.I.R., and SAOS. All seven comparisons remain significant after Holm correction across all eight baselines ($p_{\text{Holm}} \leq 8.0 \times 10^{-5}$); the 100,000-permutation run could not resolve smaller p-values. The difference between SAVE and direct Gemini 3 Flash is not statistically significant ($p_{\text{Holm}} = 0.390$).

On the 491 common error-free EgoSchema samples, SAVE achieved the highest accuracy of 75.97%, significantly outperforming VideoAgent (64.15%; $p_{\text{Holm}} = 3.25 \times 10^{-6}$) and DeepVideoDiscovery (69.04%; $p_{\text{Holm}} = 0.0052$) under paired, two-sided exact McNemar tests. Its improvements over VideoLucy (72.71%; $p_{\text{Holm}} = 0.4354$), VideoTree (73.12%; $p_{\text{Holm}} = 0.4354$), A.I.R. (73.93%; $p_{\text{Holm}} = 0.736$) and VideoCuriousAgent (72.30%; $p_{\text{Holm}} = 0.177$) were not statistically significant, however SAVE required approximately 79–83% fewer tokens than these methods. SAVE also performed comparably to both non-iterative baselines, direct Gemini 3 Flash inference (75.97% versus 75.36%; $p_{\text{Holm}} = 0.826$) and SAOS (73.12%; $p_{\text{Holm}} = 0.4354$), although both methods were more efficient, requiring 13.7K and 19.3K versus 22.5K tokens per sample respectively.

For the Gemma 4 evaluation on Minerva, SAVE achieved the highest accuracy (62.4%). In paired, two-sided permutation tests clustered by video using 100,000 permutations, SAVE outperformed all eight baselines after Holm correction. The adjusted p-value was 8.0×10^{-5} for A.I.R., VideoTree, VideoCuriousAgent, VideoAgent, direct Gemma 4 inference, DeepVideoDiscovery, and SAOS. This is the smallest Holm-adjusted value resolvable with 100,000 permutations across eight comparisons. For VideoLucy, $p_{\text{Holm}} = 0.0345$. SAVE used 19.7K tokens per question versus 72.0K for VideoLucy, a 72.6% reduction.

For Gemma on EgoSchema, SAVE achieved 72.4% accuracy, with no statistically significant difference from any of the eight baselines after Holm correction. Under paired, two-sided exact McNemar tests, SAVE performed comparably to VideoLucy ($p_{\text{Holm}} = 1.000$), direct Gemma 4 inference

($p_{\text{Holm}} = 1.000$), DeepVideoDiscovery ($p_{\text{Holm}} = 1.000$), VideoTree ($p_{\text{Holm}} = 1.000$), SAOS ($p_{\text{Holm}} = 1.000$), A.I.R. ($p_{\text{Holm}} = 1.000$), VideoAgent ($p_{\text{Holm}} = 0.168$), and VideoCuriousAgent ($p_{\text{Holm}} = 0.139$). Despite the statistically comparable performance, SAVE remained substantially more token-efficient than most prior methods, requiring only 9.4K tokens per sample compared with 34.4K for VideoTree, 103.6K for VideoLucy, 9.9K for VideoCuriousAgent, 16.0K for direct Gemma 4 inference, and 122.0K for DeepVideoDiscovery.

G SYNTHETIC EVIDENCE-LOCALITY BENCHMARK

Our benchmark contains 384 procedurally generated videos (21.6 hours in total), rendered at 256×256 resolution and 30 frames per second. Each video shows a stationary square against a white background, repeatedly changing among red, green, blue, and yellow in a fixed order. We define one complete four-color cycle as an *event*: observing a full cycle is sufficient to recover the order. Cycle duration is varied across 2, 5, 10, and 15 seconds, with each color displayed for approximately one quarter of the cycle. Video duration is varied independently across 30, 60, 120, and 600 seconds. For each video-duration and cycle-duration pair, we generate 24 videos: four instances of each of the six possible color orders anchored at red. The clip starts at a randomly selected point in the cycle, so its first visible color need not be red. Each sample asks for the order starting with red and presents five shuffled choices: the correct order and four alternatives drawn from the remaining five. Because the same event repeats throughout the video, locating the relevant event is trivial. This allows us to study the benefit of adaptive inspection independently of temporal localization. We will publicly release the videos, annotations, and generation code to support further research.

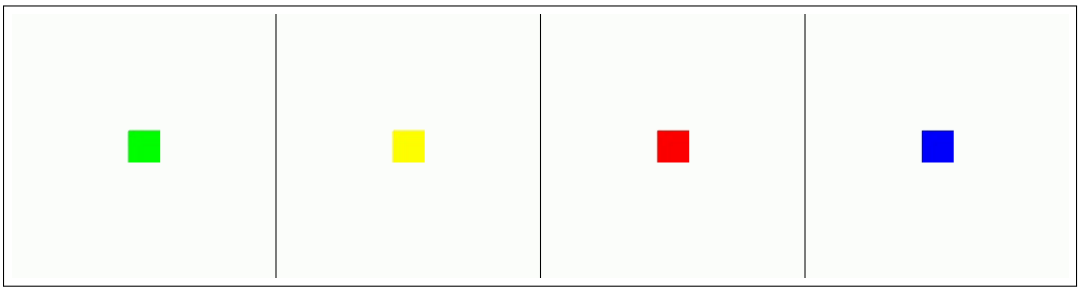


Figure 2: Illustration of our synthetic benchmark using video 3. From left to right, frames are sampled at 0, 6, 12 and 18s.

Table 3: Comparison of performance, token efficiency, and inference cost across video reasoning benchmarks and backbones.

Model	Performance (%)	Input Tokens	Output Tokens	Total Tokens	Inference Cost
Gemini 3 Flash					
Minerva					
Gemini 3 Flash Native Google DeepMind (2025)	64.8	61.1K	9.5K	70.6K	$80.54
VideoTree Wang et al. (2025c)	50.4	99.8K	6.2K	106.0K	$93.39
VideoAgent Wang et al. (2024)	38.5	**6.7K**	7.3K	**14.1K**	$34.72
VideoLucy Zuo et al. (2025)	56.5	141.1K	21.6K	162.7K	$154.13
VideoCuriousAgent Yang et al. (2025c)	51.7	97.1K	41.4K	138.5K	$235.76
DeepVideoDiscovery Zhang et al. (2025b)	52.0	68.4K	16.8K	85.2K	$30.47
A.I.R. Zou et al. (2026a)	55.0	41.0K	2.7K	43.7K	$38.95
SAOS	57.5	15.6K	**0.6K**	16.2K	**$13.12**
Active Exploration Baseline	52.8	29.4K	4.2K	33.6K	$38.08
SAVE	**66.4**	38.3K	1.5K	39.7K	$32.04
EgoSchema					
Gemini 3 Flash Native Google DeepMind (2025)	75.4	11.7K	2.0K	13.7K	$5.75
VideoTree Wang et al. (2025c)	73.1	103.4K	3.9K	107.3K	$31.13
VideoAgent Wang et al. (2024)	64.2	**2.5K**	2.8K	**5.3K**	$4.76
VideoLucy Zuo et al. (2025)	72.7	112.9K	16.6K	129.5K	$27.39
VideoCuriousAgent Yang et al. (2025c)	72.0	83.9K	21.1K	105.0K	$51.75
DeepVideoDiscovery Zhang et al. (2025b)	69.1	111.4K	13.0K	124.4K	$11.53
A.I.R. Zou et al. (2026a)	73.9	39.7K	1.9K	41.6K	$12.81
SAOS	73.1	18.8K	**0.5K**	19.3K	$5.46
Active Exploration Baseline	70.0	9.1K	1.1K	10.2K	**$3.89**
SAVE	**76.0**	21.6K	0.9K	22.5K	$6.64
VSIBench (tiny)					
Gemini 3 Flash Native Google DeepMind (2025)	**59.2**	**8.7K**	7.3K	**16.0K**	$10.47
VideoTree Wang et al. (2025c)	45.5	76.7K	4.0K	80.7K	$20.15
VideoAgent Wang et al. (2024)	40.6	10.1K	7.2K	17.2K	$10.62
VideoLucy Zuo et al. (2025)	48.2	117.0K	12.5K	129.5K	$35.37
VideoCuriousAgent Yang et al. (2025c)	45.6	101.6K	34.1K	135.6K	$61.19
DeepVideoDiscovery Zhang et al. (2025b)	51.9	29.3K	5.1K	34.5K	$5.21
A.I.R. Zou et al. (2026a)	49.9	37.1K	2.1K	39.2K	$9.88
SAOS	50.0	16.8K	**0.8K**	17.6K	**$4.28**
Active Exploration Baseline	47.2	25.8K	3.5K	29.3K	$9.40
SAVE	57.1	28.5K	1.7K	30.3K	$7.80
Gemma 4					
Minerva					
Gemma 4 Native Gemma Team (2026)	51.6	15.6K	0.3K	15.9K	
VideoTree Wang et al. (2025c)	51.6	33.0K	3.4K	36.4K	
VideoAgent Wang et al. (2024)	42.3	21.2K	1.1K	22.3K	
VideoLucy Zuo et al. (2025)	58.4	65.4K	6.6K	72.0K	
VideoCuriousAgent Yang et al. (2025c)	36.4	25.1K	1.3K	26.4K	
DeepVideoDiscovery Zhang et al. (2025b)	52.5	58.4K	13.7K	72.0K	
A.I.R. Zou et al. (2026a)	51.9	20.0K	2.1K	22.1K	
SAOS	52.5	**8.6K**	**0.3K**	**8.9K**	
Active Exploration Baseline	52.6	26.8K	3.6K	30.4K	
SAVE	**62.4**	18.9K	0.9K	19.7K	
EgoSchema					
Gemma 4 Native Gemma Team (2026)	72.4	15.7K	**0.2K**	16.0K	
VideoTree Wang et al. (2025c)	71.1	31.4K	3.0K	34.4K	
VideoAgent Wang et al. (2024)	67.7	**4.7K**	0.4K	**5.1K**	
VideoLucy Zuo et al. (2025)	73.2	89.5K	14.2K	103.6K	
VideoCuriousAgent Yang et al. (2025c)	67.7	9.3K	0.6K	9.9K	
DeepVideoDiscovery Zhang et al. (2025b)	71.5	109.9K	12.3K	122.2K	
A.I.R. Zou et al. (2026a)	**74.3**	21.1K	1.9K	23.0K	
SAOS	70.5	9.7K	0.2K	9.9K	
Active Exploration Baseline	65.3	9.4K	1.2K	10.6K	
SAVE	72.4	9.0K	0.4K	9.4K	
VSIBench (tiny)					
Gemma 4 Native Gemma Team (2026)	49.1	15.6K	0.3K	15.9K	
VideoTree Wang et al. (2025c)	42.8	22.3K	2.4K	24.6K	
VideoAgent Wang et al. (2024)	44.5	25.1K	1.1K	26.2K	
VideoLucy Zuo et al. (2025)	43.8	59.3K	4.1K	63.4K	
VideoCuriousAgent Yang et al. (2025c)	41.4	22.9K	1.3K	24.1K	
DeepVideoDiscovery Zhang et al. (2025b)	49.1	29.8K	4.8K	34.6K	
A.I.R. Zou et al. (2026a)	45.7	18.3K	1.9K	20.2K	
SAOS	43.5	**8.3K**	**0.3K**	**8.6K**	
Active Exploration Baseline	45.2	27.2K	3.8K	31.0K	
SAVE	**52.6**	10.5K	0.6K	11.1K	

H CASE STUDIES

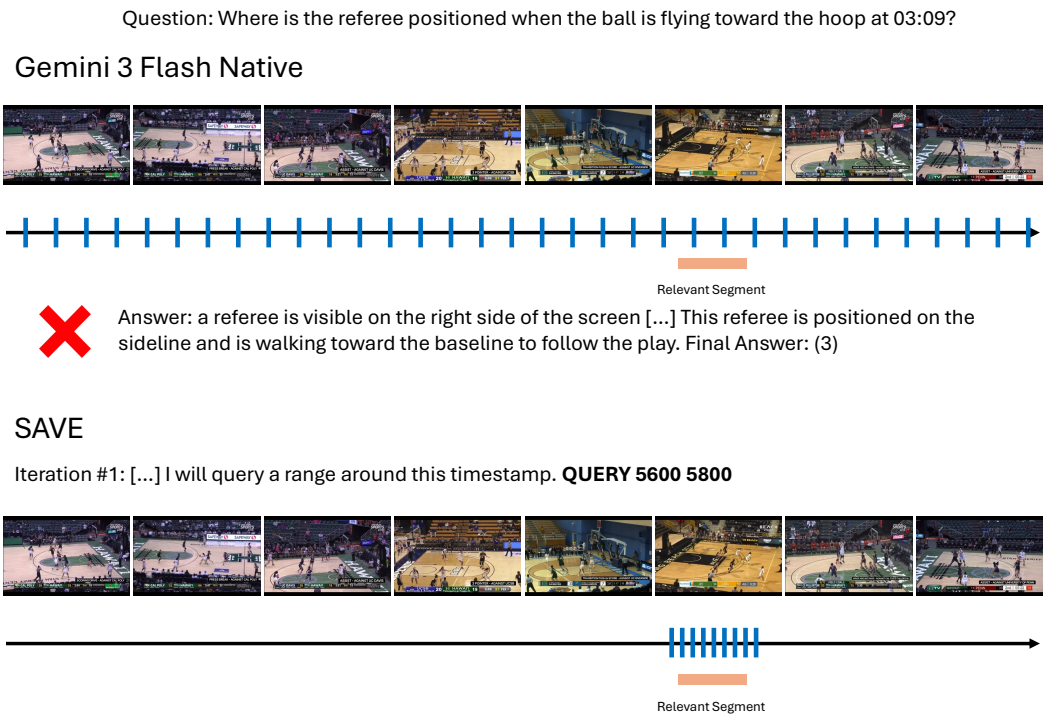
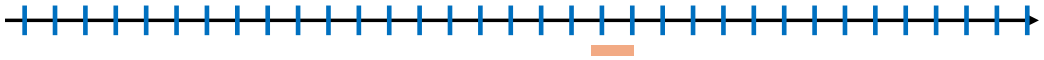


Figure 3: Example of both Gemini 3 Flash Native and SAVE answering a question from Minerva (question #7). In this example a specific timestamp is directly given in the question. Gemini simply downsamples the whole video to 1 FPS and therefore has no possibility of efficiently querying only the relevant segment. SAVE uses its range query to specifically observe the relevant part at high granularity and correctly answers the question.

Question: Where is the basketball in relation to the referee between 1:06-1:07?

Gemini 3 Flash Native



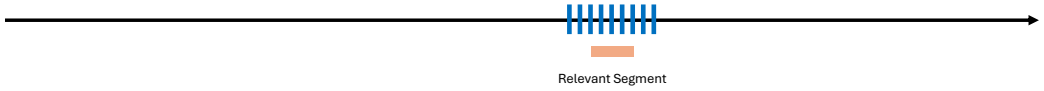


Answer: [...] From the referee's perspective, the player and the basketball are positioned in front of him and to his left. Final Answer: (3)

SAVE

Iteration #1: I need to observe the relative positions of the basketball and the referee between 1:06 and 1:07. I will query the video around this time range. **QUERY 1900 2100**





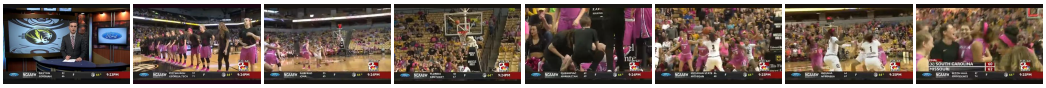
Iteration #2: [...] For a referee facing the camera, the left side of the screen is his right. Thus, the basketball is in front of him and to his right. This matches answer choice (2). **STOP**

Answer: [...] **Conclusion:** The basketball is in front of the referee and to his right. This corresponds to answer choice (2). Final Answer: (2)

Figure 4: Example of both Gemini 3 Flash Native and SAVE answering a question from Minerva (question #24). In this example a specific timestamp is directly given in the question. Gemini simply downsamples the whole video to 1 FPS and therefore has no possibility of efficiently querying only the relevant segment. SAVE uses its range query to specifically observe the relevant part at high granularity and correctly answers the question.

Question: When the banner graphic shows the second set of results, how many more points did the winning team have?

Gemini 3 Flash Native

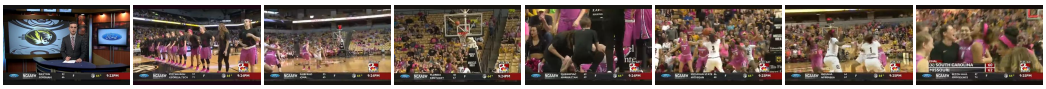


Relevant Segment

Answer: [...] show the second set of results: **NCAABW - PITTSBURGH 57, GEORGIA TECH 73** [...] The question specifically asks for the second set of results, which shows a point difference of 16. Final Answer: (3)

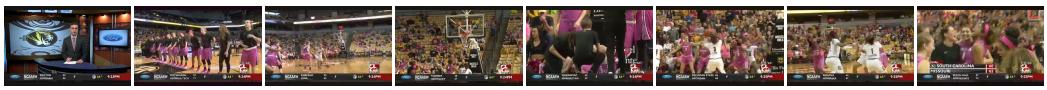
SAVE

Iteration #1: I will query the entire video at low granularity to identify the segments where banner graphics showing results appear. [...] **QUERY 0 3579**



Relevant Segment

Iteration #2: [...] the ticker at the bottom of the screen displays various NCAA women's basketball scores I need to determine which result is considered the "second set of results". The ticker cycles through games.
I will check the beginning of the video to see the sequence of results from the start to identify the first and second sets clearly. [...] **QUERY 0 1000**



Relevant Segment

Iteration #3: [...] The winning team in the second set is Georgetown with 81 points. The losing team is Xavier with 63 points. The difference is 81 - 63 = 18. This matches answer choice (1). **STOP**

Answer: [...] **Match with answer choices:** The difference of 18 points corresponds to answer choice (1). Final Answer: (1)

Figure 5: Example of both Gemini 3 Flash Native and SAVE answering a question from Minerva (question #34). No timestamp is given in the question, so both methods first have to find the relevant moment in the video. SAVE employs a multi-step search strategy: First, it queries the whole video at low granularity, subsequently isolates the crucial segment from the rest of the video and answers the question correctly. Gemini's native video processing instead passes the whole video into the model at a higher granularity than SAVE, but nonetheless fails to answer the question correctly, as fine, crucial details get lost in the large amount of irrelevant context passed into the model.

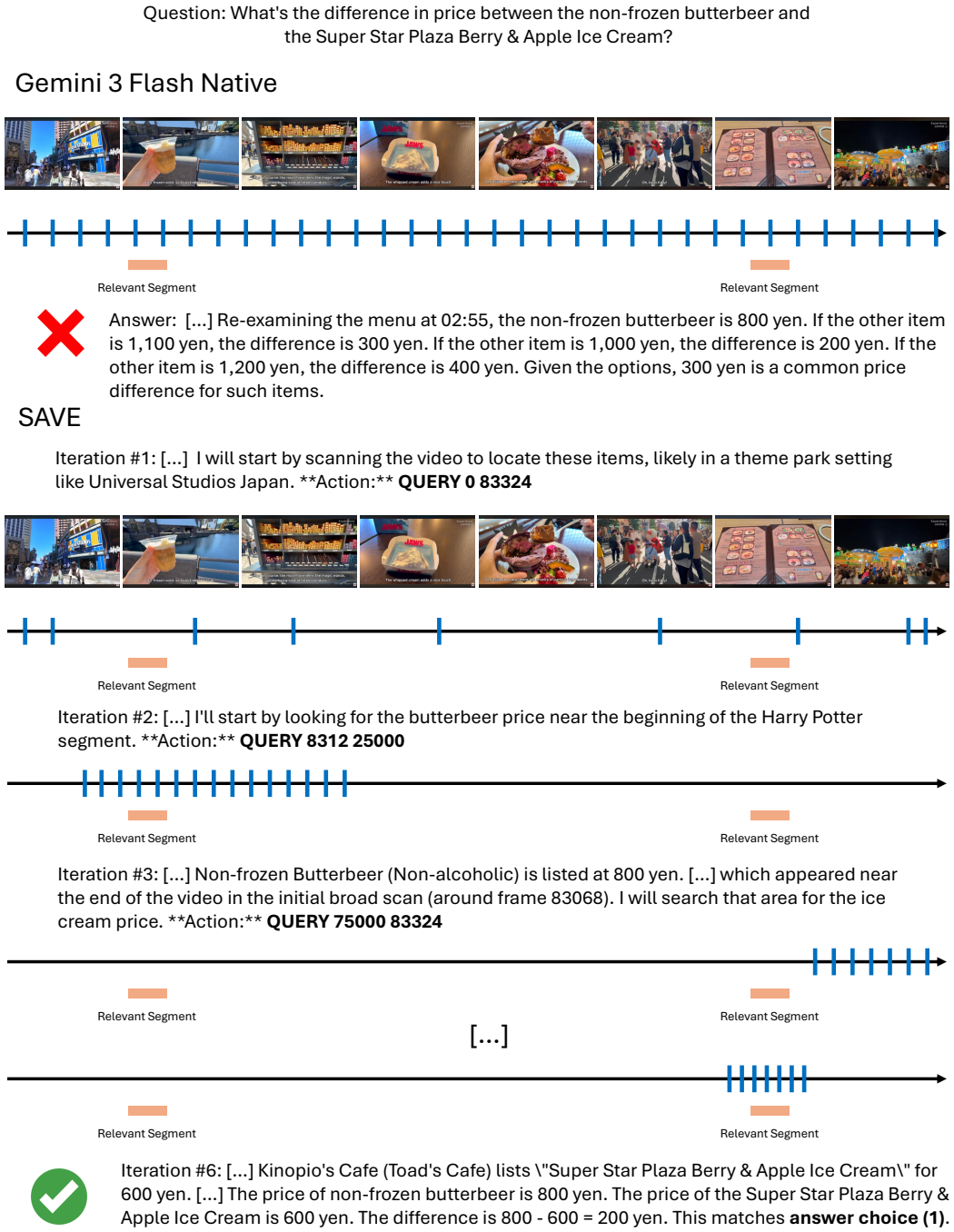


Figure 6: Example of both Gemini 3 Flash Native and SAVE answering a question from Minerva (question #1357). In this case,How two moments from the video are crucial to answering the question. Neither timestamp is given in the question. SAVE employs a multi-step search strategy: First, it queries the whole video at low granularity and subsequently isolates the first crucial segment from the rest of the video. After having correctly interpreted the first relevant segment, it proceeds to search for the second relevant segment later in the video. After 6 iterations, the second relevant moment is found and SAVE answers the question correctly. Gemini's native video processing instead passes the whole video into the model, manages to correctly identify the first moment, but the second moment is lost among the large amount of irrelevant visual context presented to the model.