DO STREAMING VIDEO MODELS REMEMBER THE PAST? VERIFYING AND EVOLVING MEMORY SYSTEM FOR VIDEOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory management is a fundamental capability for long-horizon video understanding, where models must store and retrieve useful information from a growing stream of observations. Yet, it remains unclear whether existing benchmarks actually require the memory, as even a simple memory-free system can outperform dedicated memory-augmented models on standard streaming-video benchmarks, raising questions about what these evaluations truly capture. We introduce MEMPROOF, a benchmark that verifies memory dependence by ensuring each question is unsolvable from language or recent observations alone, but becomes solvable once the necessary past evidence is revealed. With memory dependence explicitly verified, we next ask whether the memory-management strategy itself can be improved without updating any model parameters. We introduce EVOHARNESS, which keeps all neural components frozen and evolves modular programs for organizing, retrieving, and presenting memory through teacher-guided failure diagnosis and paired validation. Experiments show that strong performance on conventional streaming benchmarks can be achieved even without memory, whereas MEMPROOF requires models to actually use the past: under the same backbone, evolution improves the base harness by +8.0 points, and the memory organization and packing mechanisms of EVOHARNESS transfer to other benchmarks. The code/data will be made available online.

1 INTRODUCTION

Vision–language models (VLMs) are moving beyond short, offline video understanding toward online video, *e.g.*, long, online settings where observations arrive continuously and questions may depend on events that occurred far in the past. In this setting, memory becomes a fundamental capability: a system must efficiently preserve information over time, recover the relevant parts of the past, and integrate them with the present. As observations accumulate, retaining and processing the entire history becomes neither scalable nor necessary. Long-horizon video understanding therefore depends not simply on access to more history, but on how that history is organized, accessed, and used. A growing body of work has consequently explored memory-augmented systems for long-horizon and streaming video understanding (Zhang et al. (2025); Zeng et al. (2025); Qian et al. (2025); Yao et al. (2025)).

Despite this growing attention on memory, it remains unclear whether existing benchmarks actually require–or even measure–long-term memory. SimpleStream (Shen et al. (2026)) makes this ambiguity particularly clear: it discards long-term history altogether and feeds an off-the-shelf VLM only a few recent frames. As shown in Tab. 1, even without persistent memory, historical retrieval, or accumulated state, it outperforms existing memory-based systems on streaming video benchmarks. The behavior persists even on OVO-Bench's *Backward* subset, which is intended to evaluate historical understanding ability with questions asking about the past. This result reveals a fundamental distinction: *a question that* ***simply refers to the past*** *does* ***not mean it actually requires the past***. An earlier query-relevant event in the video may still be visible at query time, an action may be predictable from language priors, and multiple-choice questions may be guessable from semantic clues from the choices. This raises our first question:

(I) Do existing benchmarks actually require models to remember?

We argue that memory dependence should be *verified rather than assumed*. A genuine memory question should become unanswerable when the relevant past is unavailable, but answerable once available. Based on this principle, we introduce MEMPROOF, a benchmark designed to distinguish questions that merely *mention* the past from those that genuinely *depend* on it. By explicitly testing dependence on historical visual evidence, MEMPROOF provides a more direct evaluation of long-term visual memory.

Once memory dependence is established, the focus shifts from "whether a system ***needs*** *the past*" to "whether it can actually ***use*** *the past*." Current memory-augmented video methods largely follow two paradigms. One trains a model equipped with a neural memory module by updating its parameters (Zeng et al. (2025); Qian et al. (2025);

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.

Table 1: Comparison of memory-based video methods with the memory-free SimpleStream baseline (Shen et al. (2026)). SimpleStream maintains no long-term memory and observes only the four most recent frames, yet outperforms previous memory-based mechanisms on standard streaming benchmarks: StreamingBench and OVO-Bench.

Method	Venue	Memory mechanism	SBench	OVO Back	OVO Avg.
Flash-VStream (Zhang et al. (2025))	ICCV'25	Flash memory	23.2	27.4	27.9
Dispider (Qian et al. (2025))	CVPR'25	Learned latent memory	67.6	36.1	45.4
TimeChat-Online (Yao et al. (2025))	ACMMM'25	Historical token compression	75.3	41.7	51.8
StreamForest (Zeng et al. (2025))	NeurIPS'25	Persistent event memory	77.3	52.0	56.6
HERMES (Zhang et al. (2026))	ACL'26	Hierarchical KV memory	79.4	49.4	59.2
SimpleStream-4f (Shen et al. (2026))	arXiv'26	**None — last 4 frames only**	**80.6**	**54.0**	**67.7**

Yao et al. (2025)). The other keeps the neural models frozen and attaches a manually designed memory-management policy around them (Di et al. (2025); Ning et al. (2026); Zhang et al. (2026)). The former adapts neural parameters to improve memory, whereas the latter avoids training but typically keeps the memory design unchanged. In both cases, the memory-management strategy is largely specified by the designer and *remains **fixed** after all*.

Yet, memory management is itself a complex decision process: relevant evidence can be incorrectly *organized*, incompletely *retrieved*, or ineffectively *presented* to the model. Treating these decisions as ***fixed*** engineering choices leaves the previous memory system largely unoptimized. At the same time, analyzing these failures provides direct signals for how the memory system can be improved, making memory management itself a natural target of optimization. We therefore ask whether a memory system can progressively ***evolve*** while keeping the underlying neural model fixed, leading to our second question:

(II) Can the memory-management strategy be improved without updating model parameters?

We introduce EvoHarness, which treats the external memory-management strategy itself as the object of optimization. Rather than treating memory management as a fixed engineering choice, EvoHarness decomposes it into modular functions, *i.e.*, programs, that govern how past observations are organized, retrieved, and presented to a neural model. It then iteratively diagnoses failures, validates modifications, and evolves these programs to improve how the system interacts with its observed history, while keeping all the neural parameters frozen. Failure diagnoses and program-level proposals are produced by teacher LLMs, and every modification is validated against the current harness on held-out data before it is kept. Together, MemProof and EvoHarness study machine memory from two complementary directions: MemProof asks whether a task genuinely *requires* the past, while EvoHarness asks how a system can become better at *using* that past. Our experiments reveal a clear discrepancy between performance on conventional streaming benchmarks and on MemProof, where answering correctly requires demonstrable use of historical evidence. Our main contributions are summarized as follows:

- We identify a fundamental evaluation gap in streaming video understanding and introduce MemProof, a benchmark that evaluates memory through verified dependence on historical visual evidence.
- We introduce EvoHarness, which evolves the external programs for organizing, retrieving, and presenting memory through teacher-guided diagnosis and validation, while keeping all neural parameters frozen.
- Our experiments reveal a clear distinction between *having access to history* and *using memory effectively*, and show that better memory-management strategies yield substantial gains.

2 MemProof: A Long-Video Memory Benchmark

We present a new video benchmark designed around a single question: *when does a long-video question truly require memory?* The question is deceptively simple. A question that *refers* to an earlier event does not necessarily *depend* on it; the model may (i) recover the answer from language priors alone (**blind**), (ii) infer it from what is currently visible at query time (**recent**), and (iii) should not fail when all relevant past evidence is visible (**evidence**). Conventional long-/streaming-video benchmarks cannot separate these cases, because they simply record whether a model answered correctly—never why. Without checking blind, recency, and ceiling, evaluation results on such a benchmark hardly reveals whether the model actually remembered anything. We therefore build a benchmark in which a question satisfies three criteria: (i) it cannot be answered without the video, (ii) it cannot be answered from the recent frames alone, and (iii) it can be answered once the relevant past is revealed. As seen in Fig. 1, our dataset construction pipeline consists of three stages: candidate question-answer generation, memory-dependency verification, and multi-event verification.

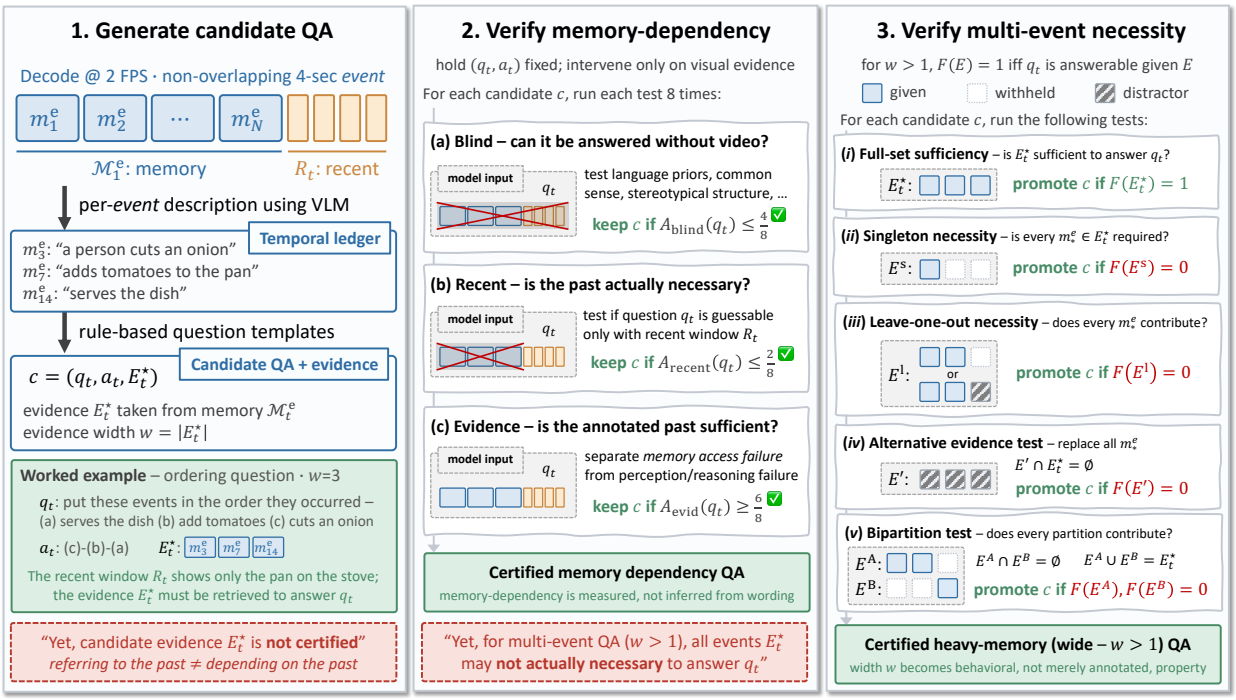


Figure 1: Overall pipeline of our benchmark construction with three main stages. We refer to Sec. 2 for details.

2.1 Candidate Question-Answer Generation

Memory-oriented questions. From a collection of videos spanning diverse long-video datasets, we start by splitting each video into a recent window and the non-overlapping events that precede it. Specifically, given a video up to query time t, $X_{\le t} = (x_1, \ldots, x_t)$, we separate the most recent observation $R_t = (x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1})$, *i.e.*, four frames sampled at 1 FPS from the four seconds preceding t, from everything that came before $\mathcal{M}_t^e = (m_1^e, \ldots, m_N^e)$, where each m_i^e is a four-second *event* before the recent window begins, satisfying $m_i^e \cap m_j^e = \varnothing \quad (i \neq j)$ and $m_1^e \cup \cdots \cup m_N^e \cup R_t = X_{\le t}$. We then caption every event $m_i^e \in \mathcal{M}_t^e$ using a VLM (Qwen3-VL-8B) and call the resulting collection of captions a *temporal ledger*: a textual record of what happens and when. From the ledger, we generate candidates with rule-based templates with an LLM conditioned on the captions of the span (detailed in supp.):

$$c = (q_t,\, a_t,\, E_t^\star), \qquad E_t^\star = \{m_{i_1}^e, \ldots, m_{i_w}^e\} \subseteq \mathcal{M}_t^e, \tag{1}$$

where q_t is the question asked at time t, a_t is its answer, and $E_t^\star$ is the past evidence the answer should come from. We call $w = |E_t^\star|$ the *evidence width*, *i.e.*, the number of temporally distinct events the answer draws on. Note that evidence width is irrelevant to video length: a long video may hold a width-one question ($w = 1$), while a short one may require combining several separated observations ($w > 1$). The evidence $E_t^\star$ records where we *expect* the answer to come from, but it does not necessarily mean that a model actually *needs* it: the question may be guessable from language alone, or answerable from R_t.

Memory-verification questions. A reliable memory system should recognize when a queried event did *not* occur. We therefore include two verification questions: (i) *Explicit verification* directly asks whether a historical claim is supported by the observed video, using balanced supported and unsupported variants as well as claim-status questions. (*e.g.*, "Did the person place the red bowl in the sink earlier?") (ii) *Implicit verification* hides the claim inside the premise of an otherwise ordinary question. For a true premise, the model must answer the requested continuation; ("After the person placed the red bowl in the sink, what happened next?") for a false premise, it must instead recognize that the stated event never occurred. Matched true/false examples share the same video, query time, and answer format, so success requires checking the premise against visual history rather than relying on question form alone.

2.2 Memory-Dependency Verification

The key step in our construction is to separate '*questions that simply* ***refer to*** *the past*' from '*questions that actually* ***depend on*** *it.*' To this end, we verify this through three stages using two independently pretrained VLM verifiers, Gemma-4-26B and Qwen3.6-35B-A3B. In the *blind* stage, we provide only the question q_t to the verifiers, without any visual input, and measure how often they recover the correct answer over repeated trials. The resulting accuracy,

$A_{\text{blind}}(q_t)$, quantifies how easily the question can be solved from language priors alone. In the *recency* stage, we provide the question together with only the recent observation R_t and count the number of correct predictions across repeated trials to obtain $A_{\text{recent}}(q_t)$. Finally, in the evidence stage, we provide the question q_t, the recent observation R_t, and the past evidence $E_t^\star$ (frames; span captions for questions defined over a continuous temporal span, see supp.). The corresponding accuracy $A_{\text{evid}}(q_t)$ measures whether the question becomes answerable once the relevant past is explicitly revealed. Thus, the verification pipeline filters candidates according to

$$\underbrace{A_{\text{blind}}(q_t) \leq \tau_{\text{blind}}}_{\text{not guessable from language}}, \quad \underbrace{A_{\text{recent}}(q_t) \leq \tau_{\text{recent}}}_{\text{not visible at query time}}, \quad \underbrace{A_{\text{evid}}(q_t) \geq \tau_{\text{evid}}}_{\text{answerable once revealed}}, \tag{2}$$

ensuring that q_t is not answerable from language or recent observations alone, but becomes answerable when the relevant past visual evidence is provided. For template-route questions, we used $(\tau_{\text{blind}}, \tau_{\text{recent}}, \tau_{\text{evid}}) = (0.5, 0.25, 0.75)$.

Verification protocol. Given $c = (q_t, a_t, E_t^\star)$, each VLM verifier predicts an answer which is compared with GT answer a_t. Each verifier repeats this procedure 4 times (*i.e.*, 8 evaluations total) to obtain a robust estimate. For multiple-choice questions, we reshuffle the choices across trials to reduce sensitivity to option ordering. For free-form responses, we avoid self-grading through cross-model evaluation: outputs generated by Gemma are judged by Qwen, and vice versa. Thus, the model producing an answer is always different from the model determining its correctness.

2.3 Multi-event Verifications

The verification above establishes that a question q_t becomes answerable when its evidence $E_t^\star = \{m^{\text{e}}_{i_1}, \ldots, m^{\text{e}}_{i_w}\}$ is provided. For a multi-event question ($|E_t^\star| = w > 1$), however, this does not guarantee that ***all*** annotated events are actually necessary (*e.g.*, q_t with $w = 3$ may still be answerable from only one or two events). We therefore apply five complementary tests to composed multi-event questions to verify whether the evidence width w reflects the true number of past observations. (i) *Full-set sufficiency*: the question must be answerable when all events in $E_t^\star$ are provided. (ii) *Singleton necessity*: we provide each event individually; if any single event is sufficient, the candidate does not require multi-event memory and is downgraded to width one ($w = 1$). (iii) *Leave-one-out necessity*: we remove each required event in turn and require the question to become unanswerable, ensuring that every annotated event contributes necessary information. (iv) *Alternative-evidence test*: we sample a disjoint evidence set $E' \subseteq \mathcal{M}_t^{\text{e}}$ such that $|E'| = |E_t^\star|$ and $E' \cap E_t^\star = \varnothing$, and reject the candidate if the same answer can be recovered from this alternative evidence. Finally, (v) *Bipartition test*: for higher-width candidates, we partition $E_t^\star$ into two non-empty subsets, $E_t^\star = E_A \cup E_B$ with $E_A \cap E_B = \varnothing$, and require both subsets to be individually insufficient. We repeat this test over multiple partitions to detect hidden lower-width solutions. For composed questions, w is thus a behaviorally verified property rather than an annotation. Questions defined over a continuous temporal span are instead certified at the span level. We refer to the supplementary material for additional details.

2.4 Final Benchmark Composition

The proposed MemProof benchmark contains 12,207 question-answer pairs over 5,437 unique videos, partitioned into 7,811 training questions over 3,281 videos and 4,396 test questions over 2,156 videos, where the two splits are strictly video-disjoint. It covers six complementary evaluation dimensions: (1) *Recent* questions are answerable from the recent observation R_t alone. (2) *Narrow* contains standard certified memory questions that require historical information beyond R_t, while (3) *Wide* targets more demanding historical reasoning over extended temporal context, including questions that require integrating information across multiple past observations. (4) *Ceiling* contains questions that remain difficult even when evidence $E_t^\star$ is directly provided, thereby separating limitations in perception/reasoning from memory access failures. (5) *Explicit* directly asks whether a claimed past event occurred, whereas (6) *implicit* embeds a potentially unsupported event inside the premise of a question, requiring the model to verify the premise.

Question diversity. Each question in MemProof varies not only in *whether* memory is required, but also in *how* information from the video should be used. The template-based generation (Sec. 2.1) spans six high-level semantic families: *ordering* (reasoning about the temporal order of separated events), *counting* (counting repeated actions/events), *enumeration* (identifying the set of relevant entities/actions), *interval reasoning* (reasoning over temporally bounded event intervals), *span verification* (verifying whether a specified event holds within a temporal span), and *state-chain reasoning* (tracking how entities evolve across multiple events), which are further organized into 15 sub-types through 111 predefined templates. These six semantic families are orthogonal to the six evaluation dimensions, *e.g.*, an ordering question may fall into different memory dimensions depending on its demand for past evidence. This separation allows MemProof to characterize both the ***reasoning operation*** required by a question and the ***memory capability*** it probes.

We refer to the supplementary material for detailed statistics of the proposed MemProof benchmark.

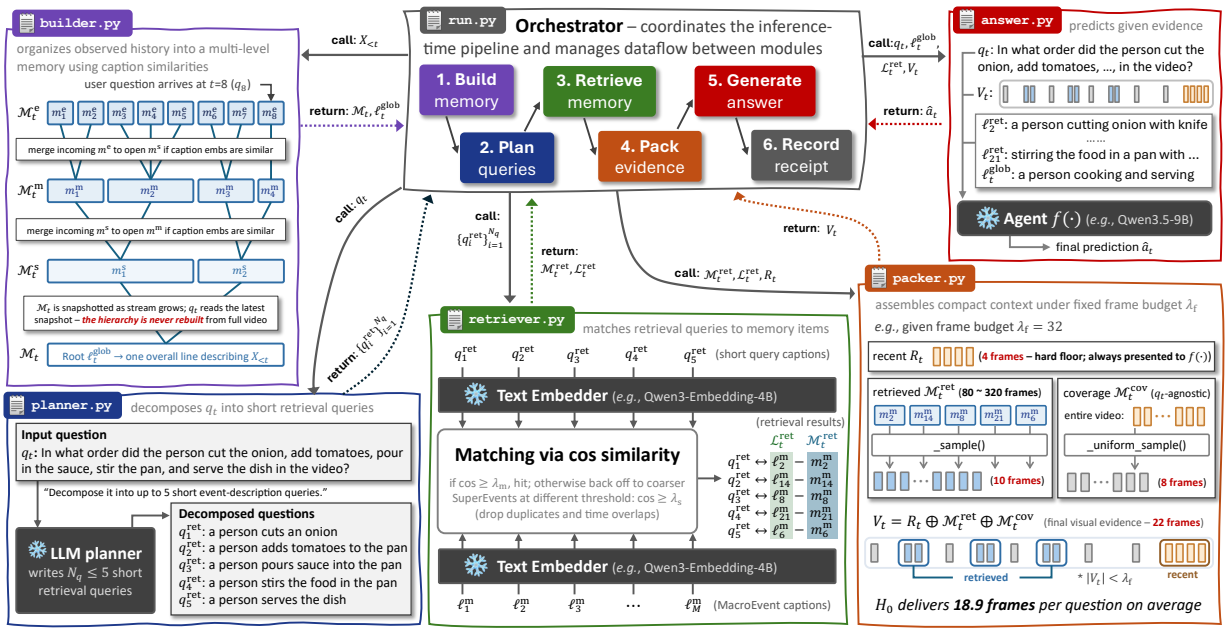


Figure 2: EvoHarness H_0. The agent $f(\cdot)$ never manages memory; external programs build, retrieve, and present it

3 EvoHarness: Iterative Evolution of Memory Harnesses

Existing memory-augmented video systems typically either finetune the VLM for a specialized memory (Zeng et al. (2025); Xu et al. (2026)), which requires *memory-specific training*, or attach a hand-designed memory policy to a frozen VLM (Di et al. (2025); Ning et al. (2026)), which leaves *the memory policy fixed*. We instead ask: "*can the memory policy itself be* ***systematically improved*** *while keeping* ***model parameters frozen****?*" Our answer is to move memory outside the model and instead ***evolve the external harness*** that governs this memory access. Starting from a base harness H_0, we progressively evolve it through repeated validation and failure diagnosis. Throughout this process, all parameters remain frozeon — only the harness evolves, *i.e.*, how to organize, retrieve, and present the history.

3.1 H_0: Base Memory Harness

Setup. We consider a video stream $X_{\le\tau} = (x_1, \ldots, x_\tau)$ that grows over time as new frames arrive. H_0 processes the stream incrementally: every four seconds, the newly observed frames form an event m_i^{e}, which is summarized by a VLM into a caption ℓ_i and appended to the temporal ledger. Thus, when a question q_t arrives at time t, the past events $\mathcal{M}_t^{\mathrm{e}} = (m_1^{\mathrm{e}}, \ldots, m_N^{\mathrm{e}})$ are represented by the ledger $\mathcal{L}_t = (\ell_1, \ldots, \ell_N)$, while the recent observation R_t remains directly available as defined in Sec. 2.1. As seen in Fig. 2, H_0 provides an interface between the agent $f(\cdot)$ and the history through three main operations: (a) memory organization, (b) memory retrieval, and (c) evidence presentation.

(a) Hierarchical memory organization. To make the growing ledger $\mathcal{L}_t$ easier to access, H_0 organizes it into a hierarchical memory. Specifically, consecutive events are grouped into a set of *MacroEvents* $\mathcal{M}_t^{\mathrm{m}} = \{m_i^{\mathrm{m}}\}_{i\in[M]}$ where $[K] \coloneqq \{1, \ldots, M\}$ for any $M \in \mathbb{Z}^+$– an incoming event m^{e} is merged into the latest MacroEvent m^{m} if their caption embeddings are sufficiently similar; otherwise, a new MacroEvent is created, where each m_i^* stores its temporal span and caption. The same procedure is applied at the MacroEvent level to form *SuperEvents* $\mathcal{M}_t^{\mathrm{s}} = \{m_i^{\mathrm{s}}\}_{i\in[L]}$ by grouping past consecutive MacroEvents. We denote the resulting hierarchical memory by $\mathcal{M}_t = (\mathcal{M}_t^{\mathrm{e}}, \mathcal{M}_t^{\mathrm{m}}, \mathcal{M}_t^{\mathrm{s}})$.

(b) Question-relevant memory retrieval. H_0 uses a frozen LLM as a *planner* (*e.g.*, Qwen3.5-9B) to decompose q_t into a set of shorter queries $\{q_i^{\mathrm{ret}}\}_{i\in[N_{\mathrm{q}}]}$ which are used for memory retrieval. Each q_i^{ret} is embedded and matched against the caption embeddings of MacroEvents in $\mathcal{M}_t^{\mathrm{m}}$ using cosine similarity. If no relevant (*i.e.*, sufficiently large similarity score) MacroEvent is found, H_0 falls back to the coarser SuperEvents in $\mathcal{M}_t^{\mathrm{s}}$ and selects the best-matching MacroEvent within the retrieved SuperEvent. We denote the retrieved visual frames (MacroEvents) by $\mathcal{M}_t^{\mathrm{ret}} \subseteq \mathcal{M}_t^{\mathrm{m}}$.

(c) Evidence presentation to the agent. In addition to the retrieved memory $\mathcal{M}_t^{\mathrm{ret}}$, H_0 sparsely samples coverage frames $\mathcal{M}_t^{\mathrm{cov}}$ across the past history to present coarse information to the agent $f(\cdot)$ beyond $\mathcal{M}_t^{\mathrm{ret}}$. Together with the recent observation R_t, the visual evidence presented to the agent is $\mathcal{V}_t = R_t \oplus \mathcal{M}_t^{\mathrm{ret}} \oplus \mathcal{M}_t^{\mathrm{cov}}$, where $\oplus$ denotes temporal concatenation. H_0 further provides a compact global textual summary ℓ_t^{glob} of the entire video and textual summaries $\mathcal{L}_t^{\mathrm{ret}}$ of the retrieved events $\mathcal{M}_t^{\mathrm{ret}}$. The agent $f(\cdot)$ then produces the final answer for q_t as $\hat{a} = f(q_t, \ell_t^{\mathrm{glob}}, \mathcal{L}_t^{\mathrm{ret}}, \mathcal{V}_t)$.

Harness as an external program. We implement these operations as external ***programs*** performing a specific operation, and collectively refer to these programs as the ***harness***. As illustrated in Fig. 2, the Memory Builder organizes the observed history, the Query Planner and Memory Retriever identify question-relevant past events, the Evidence Packer converts the retrieved events into a bounded set of visual frames, and the Answer Interface assembles the resulting evidence for $f(\cdot)$. The Orchestrator coordinates these programs and records the outputs in a receipt. Importantly, the evolution introduced next directly modifies/augments these programs, *e.g.*, adding new information to the memory, changing how retrieved events are converted into visual evidence, or altering how evidence is presented to the agent.

3.2 Iterative Harness Evolution

H_0 already provides a functional memory system, but its components may be imperfect: the planner searching for wrong events, retriever missing useful details, or correctly retrieved events being poorly presented to the agent. Our key idea is to improve these failures *without updating any model parameters* by treating the harness itself as the optimization objective through iterative evolution. At each evolution step, we evaluate the current harness H_k on the development split $\mathcal{D}_{\text{dev}}$ to perform fine-grained failure diagnosis. Based on this diagnosis, we introduce new mechanisms to each program and evaluate them on the validation split $\mathcal{D}_{\text{val}}$. Only validated mechanisms are promoted to form H_{k+1}; otherwise, H_k is retained and new proposals are explored. Starting from H_0, we repeatedly perform

$$H_k \xrightarrow{\text{evaluation on } \mathcal{D}_{\text{dev}}} \text{failure diagnosis} \rightarrow \text{new mechanisms proposal} \xrightarrow{\text{validation on } \mathcal{D}_{\text{val}}} \begin{cases} H_{k+1}, & \text{if validated,} \\ H_k, & \text{otherwise.} \end{cases} \tag{3}$$

where H_{k+1} augments H_k with new validated mechanisms. The test split is never seen throughout this evolution.

Failure diagnosis. A net evaluation score on $\mathcal{D}_{\text{dev}}$ tells us *whether* the harness H_k fails, but not *why* it fails. For fine-grained failure analysis, we record the inference trace for every example, *e.g.*, the question q_t, the retrieved memory $\mathcal{M}_t^{\text{ret}}$ and $\mathcal{L}_t^{\text{ret}}$, the final evidence presented to the agent, and its prediction $\hat{a}_t$. We refer to this inference trace as a ***receipt***. For each failed example, we compare its receipt with the annotated GT evidence $E_t^\star$ in order to pinpoint the failure: whether the evidence was not retrieved, lost before presentation, or inadequately presented to the agent. This diagnosis is performed with two LLM teachers (Claude Fable 5– Anthropic (2026) and GPT-5.6 Sol– OpenAI (2026)).

⋆ *Example evolution from H_0 to H_1.* The base harness H_0 produces 797 failures: 145 due to missed retrieval, 49 due to retrieved evidence not being properly presented, and 569 despite relevant evidence already exposed to the agent. Grouping these failures by question type reveals several recurring issues including the followings:

- **Heavy-memory question:** GT $E_t^\star$ spans a long interval, while $\mathcal{M}_t^{\text{ret}}$ contains only a narrow portion of that span.
- **Counting question:** sparsely retrieved frames may not reliably preserve how many times an event occurred.
- **Short-answer question:** text summaries $\mathcal{L}_t^{\text{ret}}$ often omit the exact object names required to answer the question.

These recurring failure patterns directly guide the design of new harness mechanisms.

New mechanism proposal. Based on the diagnosed failures, we use the two LLM teachers to generate program-level modifications to the harness. Each agent is provided with representative failure patterns with receipts to identify the cause of failures and propose potential modifications to the programs; each proposes 12 mechanisms without seeing the other's proposals. The two proposal sets are then cross-reviewed where redundant ideas are merged, and unsupported proposals are discarded, resulting in a set of modifications $\Delta_k = \{\delta_k^{(j)}\}_{j \in J_k}$ to the programs in current harness H_k.

⋆ *Example evolution from H_0 to H_1.* The diagnosed failures above lead to several direct mechanism proposals, each modifying a corresponding harness program:

- **Heavy-memory question:** the frames allocated to a MacroEvent cover only a small portion of its temporal span → `packer.py` to redistribute the same number of frames uniformly across the MacroEvent's full span.
- **Counting question:** sparse evidence obscures the number of recurring events → `builder.py` to group semantically recurring MacroEvents and maintain explicit occurrence counts during memory construction.
- **Short-answer question:** compact MacroEvent summaries omit exact object/action names → `answer.py` to additionally expose the original ledger captions associated with the retrieved MacroEvents for short-text questions.

Proposal validation. Each modification $\delta_k^{(j)}$ is implemented on top of H_k and evaluated against the original H_k on the same validation set. Since a modification may improve overall accuracy while degrading a particular capability (*e.g.*, improving counting questions while hurting heavy-memory questions) we compare the two harnesses across different evaluation groups (*e.g.*, recent, light-mem., and heavy-mem.), and reject modifications that introduce a substantial group-wise degradation. If a modification improves the net performance without substantially degrading any evaluation group, it is accepted for H_{k+1}; otherwise, H_k is retained and new modifications are explored.

$H_?$	Key changes incorporated to the external memory programs	+ / -	Δ
H_0	None; base harness	–	–
H_1	Event counts in `builder.py`; Temporal frame spreading in `packer.py`; Order-robust readout in `answer.py`	87/53	+3.66
H_2	Entity attributes in `builder.py`; Segment retrieval in `retriever.py`	33/15	+4.20
H_3	Option-mem comparison in `answer.py`	23/9	+1.09
H_4	Best-segment caption in `packer.py`; Memory-before-question in `answer.py`	81/48	+3.73
H_5	Recent-only routing in `run.py`	–	+1.26
H_6	Boundary captions in `packer.py`; Claim-specific retrieval in `planner.py` and `retriever.py`; Claim-aware comparison in `answer.py`	28/18	+1.52

Table 2: Evolution trajectory. Each row shows the main program-level changes; +/- denotes examples corrected/broken by H_k relative to H_{k-1}. Δ denotes the val accuracy gain of H_k over H_{k-1}.

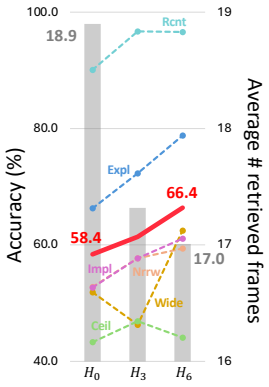


Figure 3: Acc. (lines) *vs.* # retrieved frames (bars).

4 Experimental Results and Analyses

In this section, we now investigate how EvoHarness addresses these questions: (i) how the evolution improves the harness, (ii) which memory components benefit from the evolution, and (iii) how it compares with existing systems.

Experimental setup. We develop and evaluate EvoHarness using the proposed MemProof benchmark (Sec. 2.1). We partition the MemProof training split into disjoint *development* ($\mathcal{D}_{\text{dev}}$) and *validation* ($\mathcal{D}_{\text{val}}$) subsets, which are used for the evolution process in Eq. 3. Starting from H_0, we perform six evolution iterations to obtain $H_1, \ldots, H_6$. The MemProof test split is never accessed during this process and is reserved exclusively for final evaluation. In addition, we evaluate the resulting harnesses on OVO-Bench (Niu et al. (2025)) and StreamingBench (Lin et al. (2024)) to assess their transferability to the existing streaming-video benchmarks. In all experiments, we use pretrained Qwen3.5-9B for planning and answering, Qwen3-Embedding-4B for retrieval, and Qwen3-VL-8B-Instruct for temporal ledger construction. All neural components remain frozen throughout harness evolution; only the external memory harness is modified. Detailed model configurations, data partitions, and evolution protocols are provided in the supplementary.

4.1 Harness Evolution From H_0 to H_6

We start from the base H_0 and iteratively evolve its external memory programs through validation-driven updates. We observed that performance improves consistently over the course of evolution and largely converged after H_6.

Program-level evolution. Table 2 illustrates how EvoHarness improves the memory system by repeatedly diagnosing failures and modifying the responsible programs. At each iteration, the current harness is first executed on a development subset $\mathcal{D}_{\text{dev}}$, where teacher agents inspect its errors and inference receipts to identify failure modes and propose program-level fixes. The proposed changes are then evaluated on the validation data $\mathcal{D}_{\text{val}}$, and only those that improve the overall validation performance without causing substantial regressions are promoted to the next harness.

In the evolution stage from H_3 to H_4 as an example, the failure diagnosis leads the teacher agents to focus on how retrieved evidence (*e.g.*, $\mathcal{M}_t^{\text{ret}}$ and $\mathcal{L}_t^{\text{ret}}$) should be delivered to $f(\cdot)$, rather than simply retrieving more history from $\mathcal{M}_t$; H_3 may successfully retrieve the right MacroEvent, but still present the agent $f(\cdot)$ with unreliable evidence. Specifically, retrieved MacroEvents cover several smaller events, so its caption summary does not highlight the specific moment needed to answer q_t. The teachers therefore modify `packer.py` to additionally provide the caption of the most relevant fine-grained event inside each retrieved MacroEvent, also modifying `answer.py` to place the retrieved memory immediately before q_t in the final prompt. These changes together make the already-retrieved evidence easier for the agent $f(\cdot)$ to use, improving the six-item objective on $\mathcal{D}_{\text{val}}$ from 58.0 to 61.8. In paired comparison, H_4 corrects 81 examples that H_3 incorrectly answered while breaking 48 previously correct examples, yielding a net gain of +33. Other evolution steps follow the same diagnose–modify–validate procedure while keeping all neural models frozen. We provide the complete evolution trajectory of Tab. 2 in the supplementary material.

Accuracy *vs.* frame budget. Figure 3 shows the evolution at some representative checkpoints H_0, H_3, and H_6. The red line denotes overall accuracy on MemProof *test set*, while the dashed lines show accuracy on the six evaluation dimensions. Although some individual dimensions temporarily decreased during evolution, we promote a harness based on its overall validation improvement, while assuring it does not introduce a substantial regression in any single dimension. The six-item objective increases by 13.7% (58.4 → 66.4), while the average number of retrieved frames decreases by 10.1% (18.9 → 17.0). The results supports that the improvement comes not from providing more visual evidence, but from using a *smaller* amount of *more relevant* evidence with *more effective* (evolved) programs.

	System	Venue	Memory	MemProof							OVO-Bench		SBench	MME	LongVB
				Rcnt	Nrrw	Wide	Ceil	Expl	Impl	All.	Back	Avg.	Avg.	Long	Val.
Finetuned	TimeChat-Online (Yao et al. (2025))	MM'25	token drop	91.5	51.8	**54.7**	26.9	51.1	**33.7**	50.0	41.7	51.8	75.3	**52.4**	57.1
Finetuned	StreamForest (Zeng et al. (2025))	NeurIPS'25	event tree	87.7	46.3	44.4	21.4	42.1	29.4	42.6	**52.0**	56.6	77.3	–	–
Finetuned	Dispider (Qian et al. (2025))	CVPR'25	latent state	83.9	35.8	38.3	31.0	23.6	18.9	35.4	–	–	67.6	–	–
Finetuned	Mage-VL (Yang et al. (2026))	arXiv'26	N/A	92.3	**53.4**	50.9	**37.2**	55.3	31.6	**52.3**	48.2	**64.0**	–	–	**61.3**
Finetuned	VideoChat3 (Li et al. (2026))	arXiv'26	token compress	**94.3**	45.2	54.5	35.4	**58.1**	29.1	51.4	–	–	**83.0**	–	–
Frozen	ReKV (Di et al. (2025))	ICLR'25	KV retrieve	88.0	50.3	40.5	43.7	49.6	28.5	49.4	44.2†	50.8†	69.2†	–	–
Frozen	InfiniPot-V (Kim et al. (2025))	NeurIPS'25	KV compress	86.9	55.5	50.0	39.4	52.5	36.9	52.6	47.6	56.8	76.4	53.4	58.4
Frozen	HERMES (Zhang et al. (2026))	ACL'26	memory bank	91.9	44.4	17.2	41.8	41.9	27.8	43.2	49.4	59.2	79.4	53.4	–
Frozen	FluxMem (Xie et al. (2026))	CVPR'26	token compress	87.5	46.8	51.6	38.7	51.0	31.7	50.1	47.2†	57.2†	76.4	54.0	61.1
Frozen	SimpleStream-4f (Shen et al. (2026))	arXiv'26	window	87.2	17.8	2.4	17.9	37.9	36.4	30.7	48.7†	62.4†	79.0†	–	–
Frozen	CausalMem (Song et al. (2026))	arXiv'26	memory bank	88.7	48.9	44.7	41.4	44.6	21.2	47.6	–	–	74.3	–	57.0
	EvoHarness H_6	**ours**	evolved harness	96.6	**59.4**	**62.5**	44.1	**78.8**	**61.1**	**66.4**	**57.1**	64.9	79.3	55.3	**63.1**
	EvoHarness H_3	**ours**	evolved harness	**96.7**	57.7	46.3	**46.9**	72.3	57.7	61.4	57.0	**65.9**	**80.5**	**56.1**	62.3

Table 3: Method comparison on MemProof, streaming (OVO-Bench, StreamingBench), and offline long-video benchmarks (VideoMME, LongVB). External systems use their own backbones. MemProof **All** is the six-item objective. † Reproduced with our Qwen3.5-9B backbone. (More results are in supp.) Bold/underline: best/second.

4.2 Method Comparison & Transferability Across Video Benchmarks

Table 3 compares EvoHarness with existing "memory-augmented" video systems across MemProof, conventional streaming benchmarks, and offline long-video benchmarks. We evolve the harness exclusively on the MemProof training split and evaluate it on the held-out MemProof test split. The evolved harness is then transferred directly to the streaming and offline long-video benchmarks without any target-benchmark-specific adaptation, allowing us to assess *whether MemProof-evolved* ***memory strategies generalize*** *across different benchmarks* beyond itself.

Overall comparison. With all neural components frozen, EvoHarness H_6 reaches 66.4 on MemProof. External systems use different backbones and are evaluated zero-shot, so we attribute gains with same-backbone controls (supp.): a 68-frame uniform control already reaches 56.4, *i.e.*, 3.8 points over the strongest external system (52.6) stem from backbone and pipeline; H_0 is paired-tied with this control (net $+3$ questions), and evolution adds $+8.0$ (H_6 *vs.* H_0).

MemProof removes recency shortcuts. Table 1 shows that a simple memory-free baseline (Shen et al. (2026)) outperforms prior memory-augmented methods on streaming-video benchmarks, revealing an evaluation gap: "strong performance on those does not necessarily imply effective memory use." Table 3 makes this distinction clear. While SimpleStream remains strong on conventional streaming benchmarks, its accuracy drops to 30.7 on MemProof and to only 2.4 on Wide (memory-heavy questions), exposing its inability to effectively use long-term memory. In contrast, EvoHarness H_6 achieves 66.4 overall and 62.5 on Wide. This collapse is expected by construction and confirms that our filters remove recency shortcuts; the more informative comparison is against same-backbone uniform sampling of the past, which H_6 exceeds most on Wide (refer to the supplementary for the experimental results).

The dimension-wise results further show that MemProof effectively separates short-horizon perception from long-horizon memory demand. On Recent, where memory is unnecessary, H_3 and H_6 perform almost identically (96.7 *vs.* 96.6). In contrast, the evolution ($H_3 \rightarrow H_6$) improves on memory-demanding dimensions such as Narrow (57.7 $\rightarrow$ 59.4), Wide (46.3 $\rightarrow$ 62.5), Explicit (72.3 $\rightarrow$ 78.8), and Implicit (57.7 $\rightarrow$ 61.1). The largest gain appears on Wide, where $H_3 \rightarrow H_6$ improves from 46.3 to 62.5. Ceiling does not improve (46.9 $\rightarrow$ 44.1), as expected when evidence alone is insufficient. MemProof thus separates what a system perceives now from what it retrieves from the past.

Evaluation on streaming-/offline long-video benchmarks. The transfer results reveal three complementary findings about how our memory strategy behaves: (1) On OVO-Bench *Backward*, which explicitly demands past, EvoHarness outperforms prior frozen memory systems (*e.g.*, ReKV, InfiniPot-V, HERMES, FluxMem). Note that they primarily tackle *how to* ***retain and compress*** *the history*, whereas EvoHarness ***additionally*** focuses on *how to* ***organize, retrieve, and present*** *the history*. (2) On StreamingBench, which places greater emphasis on real-time/short-horizon understanding, EvoHarness remains competitive despite never being evolved on this benchmark. Unlike OVO-Bench *Backward*, strong performance here does not necessarily require long-term memory; accordingly, the more memory-specialized H_6 does not uniformly outperform H_3. (3) Although *our harness is evolved only on* ***streaming-style*** MemProof and transferred without any offline-benchmark adaptation, EvoHarness outperforms prior frozen memory systems in offline setting as well. Transfer gains come mainly from the memory-organization and packing mechanisms of $H_0 \rightarrow H_3$; the verification-oriented mechanisms of $H_3 \rightarrow H_6$ chiefly benefit MemProof (refer to the supp. for details), indicating reusable strategies rather than benchmark-specific heuristics in the former.

Do the evolved memory mechanisms transfer across backbones? We further test whether the improvements learned during harness evolution are specific to the backbone. As shown in Table 4, the transition from H_3 to H_6 yields nearly the same improvement with two different backbones: +4.97 points with Qwen3.5-9B and +4.49 points with Qwen3-VL-8B, corresponding to 234 and 202 net corrected examples, respectively. Thus the $H_3 \rightarrow H_6$ (verification) mechanisms are not tied to one answering model. This is not observed for $H_0 \rightarrow H_3$: with Qwen3-VL-8B, H_0 already scores 58.3 (supp.), so organization/packing gains transfer across benchmarks but not clearly across backbones.

	Q3.5-9B	**Q3-VL-8B**
H_3	61.41	58.59
H_6	**66.38**	**63.08**
Δ	**+4.97**	**+4.49**
net	+234	+202

Table 4: MemProof test results of H_3 and H_6 with different answering backbones $f(\cdot)$.

5 Related Work

Long- and Streaming-Video Benchmarks. Recent benchmarks have extended evaluation from short clips to longer and temporally richer videos. EgoSchema (Mangalam et al. (2023)) evaluates long-form egocentric understanding and introduces temporal certificate sets to characterize the temporal extent needed to answer. MLVU (Zhou et al. (2025)), Video-MME (Fu et al. (2025)), LongVideoBench (Wu et al. (2024)), and LVBench (Wang et al. (2025)) further increase duration, task diversity, and required temporal context. Complementarily, StreamingBench (Lin et al. (2024)) and OVO-Bench (Niu et al. (2025)) move evaluation to *online* settings, where questions arrive during observation.

While these benchmarks establish challenging settings for long-context visual understanding, most are not designed to isolate *memory dependence* itself; Long duration and annotated timestamps indicate where useful information may occur, but do not establish that past information is actually required, as questions may be solvable from language priors or recent observations alone. HERBench (Ben-Ami et al. (2026)) comes closest by enforcing multi-evidence integration, but only in offline VideoQA rather than causal memory dependence in a streaming setting. In this work, the proposed MemProof explicitly ensures that a question is unanswerable without the video or from recent observations alone, but becomes answerable once the specific past evidence is revealed, complementing existing video benchmarks.

Memory-Augmented Video Understanding. Long videos have motivated a broad range of memory-augmented systems. Flash-VStream (Zhang et al. (2025)) maintains dedicated context and augmentation memories, while StreamForest (Zeng et al. (2025)) organizes history into a event-memory hierarchy. ReKV (Di et al. (2025)) stores and retrieves query-relevant KV caches. Other approaches (Yao et al. (2025); Kim et al. (2025); Zhang et al. (2026); Xie et al. (2026); Song et al. (2026)) primarily apply *compression* to manage growing visual or KV states.

These methods make important progress on *how to retain, compress, or retrieve* an ever-growing video history. Yet, the benchmark limitation discussed above makes it difficult to determine whether overall net score actually reflects a better memory mechanism. Accordingly, the strong performance of the memory-free SimpleStream baseline (Shen et al. (2026)) on existing streaming benchmarks makes this concern particularly clear. Moreover, existing systems generally either (i) *finetune a neural memory-module* or (ii) *manually design a fixed memory policy*. Our work takes a different direction: using the verified GT evidence in MemProof, EvoHarness diagnoses where memory use fails and *iteratively improves the external policy*, while keeping neural models frozen.

Evolving Agent Harnesses. A harness is the external code that determines what information a model stores, retrieves, and receives as input (Lee et al., 2026; Huang et al., 2026). Recent work uses execution feedback to improve harness code or agent memory designs (Lee et al., 2026; Huang et al., 2026; Xiong et al., 2026). Related approaches have also explored evolving retrieval strategies and agent programs for video understanding (Ji et al., 2026; Cui et al., 2026; Xu & Chen, 2026). In our work, MemProof's verified historical evidence provides feedback for EvoHarness to improve how video history is managed. We provide a detailed comparison with these concurrent methods in supp.

6 Conclusion

We studied memory for videos from two coupled perspectives: *whether a task truly requires the past*, and *whether a system can improve how it uses that past*. Building on MemProof, which turns memory demand into a verifiable property, EvoHarness constructs an external memory and iteratively improves how that memory is used. Together, they form a closed loop between identifying memory failures and improving the strategies that cause them. We believe this perspective points to a broader direction for long-horizon AI systems. As visual streams grow longer, simply enlarging context windows or retaining more states becomes increasingly impractical; the central challenge is deciding what information should remain accessible, when it should be recalled, and how it should be presented for the current decision. Our results suggest that these decisions can themselves be improved through an external, evolving policy, reframing memory from a fixed component into a capability that can continually adapt with experience.

REFERENCES

Anthropic. Claude fable 5 and claude mythos 5. https://www.anthropic.com/news/claude-fable-5-mythos-5, 2026. 6

Dan Ben-Ami, Gabriele Serussi, Kobi Cohen, and Chaim Baskin. Herbench: A benchmark for multi-evidence integration in video question answering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 9

Benlei Cui, Ruize Wang, Junjie Li, Jinhao Chen, Longtao Huang, Yinghao Chen, Yuwen Zhai, Jingqun Tang, Ruijian Jia, Weiwei Wu, Pengfei Sun, and Haiwen Hong. MetaVideoAgent: Automated video-agent evolution for long-form video understanding, 2026. URL https://arxiv.org/abs/2608.04587. 9

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video kv-cache retrieval. In *International Conference on Learning Representations (ICLR)*, 2025. 2, 5, 8, 9

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 *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 9

Yue Huang, Wenjie Wang, Han Bao, Yuchen Ma, Xiaonan Luo, Yi Nian, Haomin Zhuang, Zheyuan Liu, Yue Zhao, and Xiangliang Zhang. MemoHarness: Agent harnesses that learn from experience, 2026. URL https://arxiv.org/abs/2607.14159. 9

Yixin Ji, Fanghua Ye, Juntao Li, Bo Zhao, Zexuan Qiu, Zhaopeng Tu, Liefeng Bo, and Min Zhang. Homer: Understanding long-form videos with hierarchical memory and agentic reasoning, 2026. URL https://arxiv.org/abs/2607.02588. 9

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. Infinipot-v: Memory-constrained kv cache compression for streaming video understanding. 2025. 8, 9

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052. 9

Xinhao Li, Yuhan Zhu, Xiangyu Zeng, Yuhao Dong, Haoning Wu, Zhiqiu Zhang, Yuandong Yang, Changlian Ma, Qingyu Zhang, Yansong Shi, Xinyu Chen, Haoran Chen, Zizheng Huang, Jun Zhang, Kun Ouyang, Lin Sui, Ziang Yan, Yicheng Xu, Chenting Wang, Yinan He, Hongjie Zhang, Yi Wang, Yu Qiao, Yali Wang, Ziwei Liu, Kai Chen, and Limin Wang. Videochat3: Fully open video mllm for efficient and generalist video understanding. *arXiv preprint arXiv:2607.14935*, 2026. 8

Junming Lin, Zheng Fang, Chi Chen, Zihao Wan, Fuwen Luo, Peng Li, Yang Liu, and Maosong Sun. Streamingbench: Assessing the gap for mllms to achieve streaming video understanding. *arXiv preprint arXiv:2411.03628*, 2024. 7, 9

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. 2023. 9

Zhenyu Ning, Guangda Liu, Qihao Jin, Chengwei Li, Wenchao Ding, Minyi Guo, and Jieru Zhao. Livevlm: Efficient online video understanding via streaming-oriented kv cache and retrieval. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2026. 2, 5

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. Ovo-bench: How far is your video-llms from real-world online video understanding? In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 7, 9

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, 2026. 6

Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video llms with active real-time interaction via disentangled perception, decision, and reaction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2, 8

Yujiao Shen, Shulin Tian, Jingkang Yang, and Ziwei Liu. A simple baseline for streaming video understanding. *arXiv preprint arXiv:2604.02317*, 2026. 1, 2, 8, 9

Baiyang Song, Yuli Lin, Qiong Wu, Tao Chen, Jun Peng, Xiao Chen, Yiyi Zhou, and Rongrong Ji. Towards a dynamic and fixed-budget memory bank for efficient streaming video understanding. *arXiv preprint arXiv:2606.25658*, 2026. 8, 9

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 *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2025. 9

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. 2024. 9

Yiweng Xie, Bo He, Junke Wang, Xiangyu Zheng, Ziyi Ye, and Zuxuan Wu. Fluxmem: Adaptive hierarchical memory for streaming video understanding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 8, 9

Yiming Xiong, Shengran Hu, and Jeff Clune. Learning to continually learn via meta-learning agentic memory designs, 2026. URL https://arxiv.org/abs/2602.07755. 9

Guoyang Xu and Hao Chen. VideoHarness-RSI: Recursive harness self-improvement for long-video understanding with frozen vision-language models, 2026. URL https://arxiv.org/abs/2608.24302. 9

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. Streamingvlm: Real-time understanding for infinite video streams. In *International Conference on Learning Representations (ICLR)*, 2026. 5

Senqiao Yang, Kaichen Zhang, Zhaoyang Jia, Jinghao Guo, Yifei Shen, Xinjie Zhang, Xiaoyi Zhang, Haoqing Wang, Xiao Li, Peng Zhang, Xiang An, Yin Xie, Zhening Liu, Xun Guo, Jiahao Li, Shicheng Zheng, Jinglu Wang, Zongyu Guo, Wenxuan Xie, Zihan Zheng, Yuxuan Luo, Bin Li, and Yan Lu. Mage-vl: An efficient codec-native streaming multimodal foundation model. *arXiv preprint arXiv:2607.24904*, 2026. 8

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, Lingpeng Kong, Qi Liu, Yuanxing Zhang, and Xu Sun. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proc. ACM Multimedia Conference (ACMMM)*, 2025. 1, 2, 8, 9

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, Yi Wang, and Limin Wang. Streamforest: Efficient online video understanding with persistent event memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 1, 2, 5, 8, 9

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-vstream: Efficient real-time understanding for long video streams. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2025. 1, 2, 9

Haowei Zhang, Shudong Yang, Jinlan Fu, See-Kiong Ng, and Xipeng Qiu. Hermes: Kv cache as hierarchical memory for efficient streaming video understanding. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. 2, 8, 9

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: Benchmarking multi-task long video understanding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 9