PIVOT: A Unified Agentic Framework for Streaming Long-Video Understanding

Qiushi Lyu **Qianlan Yang** **Ziqi Pang** **Yu-Xiong Wang** **Liang-Yan Gui**
University of Illinois Urbana-Champaign
{qlyu2,qianlan2,ziqip2,yxw,lgui}@illinois.edu

Abstract

Streaming long-video understanding is a realistic setting for always-on video assistants, where visual input arrives continuously and users may ask questions at arbitrary times. We study streaming long-video question understanding (QA) under a unified temporal formulation in which multiple questions are revealed over time and may concern the past, present, or future relative to their query times. The model receives no task-type labels and must operate under strict sequential access and bounded shared memory, deciding whether each active question is already answerable or should remain open for future evidence. This differs from prior streaming settings that often focus on short streams, prefix-answerable queries, separated temporal categories, or response timing without long-horizon shared memory. To evaluate this formulation, we propose Ref2Stream (*Reference-to-Stream*), a general benchmark-conversion protocol that turns temporally grounded offline video QA benchmarks into streaming QA benchmarks. Instantiated on LVBench, Ref2Stream yields LVBench-Ref2Stream, which mixes past, present, and future questions under strict sequential access and evaluates both answer accuracy and answer latency. Existing streaming video methods typically improve efficiency or response timing through predefined compression, retrieval, or response policies, while offline video agents perform adaptive evidence gathering but assume access to the full video and target question. We introduce PIVOT (***P**lanning over **I**ncremental **V**ideo **O**bservations for **T**imely Answering*), a unified agentic framework that incrementally updates bounded memory from the observed stream and uses a planner-controlled action loop to retrieve relevant memory, analyze current evidence, reflect on candidate answers, and decide whether to answer or defer. Our work suggests that agentic reasoning is a promising direction for building streaming long-video systems that can adaptively gather evidence and answer only when sufficient information has been observed. We release our code and benchmark at: https://github.com/AlfredLyu/PIVOT.

1 Introduction

Video understanding is increasingly moving from offline analysis toward online, streaming scenarios. In many real-world applications, such as surveillance, autonomous driving, augmented reality, robotics, and assistive systems, visual inputs arrive continuously over time and decisions must be made online. This differs fundamentally from conventional offline video understanding [15, 16, 17, 25], where a model can inspect the entire video and answer questions after all frames have been observed. In a streaming setting, the model must process frames sequentially, maintain useful information from the past, and decide whether a question can be answered from the evidence observed so far.

Existing video understanding settings [2, 9, 10, 15] only partially capture this streaming requirement. Offline long-video question answering (QA) benchmarks evaluate reasoning over extended temporal context, but they assume access to the full video and usually reveal the question after the video

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

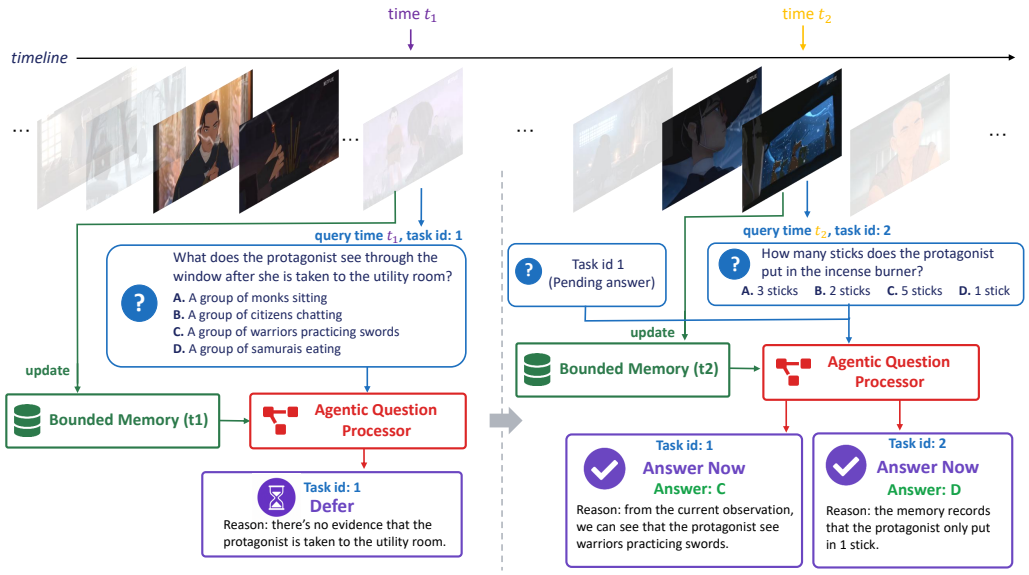

Figure 1: *Overview of our streaming setting*. A video is observed as a continuous stream, while questions are revealed online at their query times. Questions may be past-, present-, or future-oriented with respect to their query time. The first question, asked at t_1, is future-oriented: its evidence has not appeared at its asking time, so the desired action is to defer until the relevant scene is observed at time t_2 and then answer. The second question, asked at t_2, is past-oriented: its evidence has already occurred (at previous highlighted frames), so the system should answer immediately. PIVOT realizes this behavior by maintaining bounded memory for crucial visual details and using an agentic question processor to decide whether each active question should be answered now or deferred.

has been observed. Recent streaming benchmarks take a step toward this goal by posing questions at intermediate timestamps and evaluating temporally situated capabilities, such as recalling past events, understanding ongoing visual content, or waiting for future evidence [9, 10]. However, these settings often emphasize only a subset of the streaming problem: some focus on questions answerable from the observed prefix, some separate past-, present-, and future-oriented queries into task-specific categories, and some study response timing without requiring multiple long-video questions to share a bounded streaming memory. As a result, they do not fully test whether a model can handle long video streams where multiple questions arrive over time, share a bounded memory, and require reasoning over past memory, current perception, and future evidence under the same protocol.

In this work, we study *streaming long-video understanding* in a more systematic way. As illustrated in Figure 1, this setting reflects realistic always-on video assistants, where visual input arrives continuously, users may ask questions at arbitrary times, and the system must answer using only what it has observed and remembered so far. Our setting is designed around three key requirements. First, we consider a *unified temporal setting*: questions may be about the past, present, or future relative to their query time, and the model is not given the question type. Thus, the model cannot rely on oracle routing that tells it whether a question should be answered immediately or held open for future evidence. Instead, it must determine from the observed stream whether the question is already answerable or whether answering now would be premature. Second, we impose a *bounded shared memory constraint*: since the video stream can be long and multiple questions may arrive over time, the model cannot store all past frames or optimize memory only for known questions. It must maintain a compact, query-agnostic memory that can support future queries as well as currently active ones. Third, we focus on *long videos*, where questions can depend on evidence that appears far apart in time. This makes the setting substantially different from short streaming clips and places stronger demands on memory, retrieval, and answer timing. Our contribution lies in the joint setting rather than any individual ingredient: hiding temporal-type labels makes type inference part of the task, while bounded shared memory requires retaining evidence before future questions are known.

To evaluate this formulation, we propose Ref2Stream, a representative benchmark-conversion protocol for transforming temporally grounded offline video QA benchmarks into streaming QA benchmarks.

And we instantiate Ref2Stream on LVBench [15], an offline long-video understanding benchmark with reference time intervals, resulting in LVBench-Ref2Stream. LVBench-Ref2Stream mixes past, present, and future questions under the same streaming protocol, enabling evaluation of both answer accuracy and answer latency under strict sequential access and bounded memory.

This setting also exposes limitations of existing modeling approaches. Prior streaming video methods typically address online processing through memory compression, KV-cache retrieval, token pruning, response-state prediction, or readiness-aware objectives [1, 2, 4, 6, 13, 18, 19, 21, 24]. These methods improve efficiency or response timing, but they typically rely on predefined compression, retrieval, or response policies rather than question-specific planning over evolving evidence. In contrast, recent video agents show that Large Language Model planners can iteratively gather evidence, use tools, and decide when enough information has been collected for offline long-video understanding [12, 16, 17, 20, 22, 25]. However, directly applying offline agents to streaming videos is not possible, because they typically assume access to the full video and the target question before inference.

Therefore, as illustrated in Figure 1, we introduce PIVOT (***P**lanning over **I**ncremental **V**ideo **O**bservations for **T**imely Answering*), a unified agentic framework for streaming long-video understanding. PIVOT incrementally processes the observed video stream and separates streaming memory maintenance from question-specific reasoning. As new visual observations arrive, it updates a bounded memory using only the observed content; for each active question, a planner-controlled action loop retrieves relevant memory, analyzes current evidence, reflects on candidate answers, and decides whether to answer or defer. By making evidence gathering and answer timing planner-controlled decisions, PIVOT handles past, present, and future questions within the same framework without task-type labels.

Our contributions are as follows: (1) We formulate streaming long-video understanding as a unified temporal setting where models must handle past, present, and future questions without task-type labels, under strict sequential access and bounded shared memory. (2) We propose Ref2Stream and instantiate it on LVBench to construct LVBench-Ref2Stream, a streaming long-video QA benchmark with query-time and answer-time annotations for evaluating answer accuracy and answer latency. (3) We introduce PIVOT, a modular agentic framework in which a planner coordinates memory retrieval, question analysis, reflection, and answer/defer decisions over bounded streaming memory.

2 Related Work

Streaming video understanding and benchmarks. Streaming video understanding requires models to process frames sequentially, retain useful history, and respond online. Prior methods improve different aspects of this pipeline: VideoLLM-online [2] converts offline temporal annotations into streaming dialogue; Flash-VStream [24], ReKV [4], StreamMem [19], InfiniPot-V [6], and StreamForest [23] study efficient streaming memory through dual-process designs, KV-cache retrieval/compression, or event-level memory; and Dispider [13], TimeChat-Online [21], Streamo [18], and StreamReady [1] focus on interaction efficiency and response timing. Recent benchmarks such as StreamingBench [9] and OVO-Bench [10] further evaluate temporally situated online video understanding, including backward tracing, real-time perception, and forward responding. However, existing methods and benchmarks typically emphasize short streams, prefix-answerable queries, separate temporal categories, memory efficiency, or response timing. More specifically, OVO-Bench [10] separates temporal modes into task groups, StreamingBench [9] uses full-prefix evaluation for non-Proactive-Output tasks, and VideoLLM-online [2] uses type-specific query templates. StreamReady [1] studies proactive answer readiness rather than unified past/present/future QA. In contrast, our setting evaluates long-video streams with multiple questions revealed over time, requiring unified past, present, and future reasoning without task-type labels under strict sequential access and bounded shared memory.

Video agents. Recent video agents use Large Language Models as planners to selectively gather evidence for long-video understanding. VideoAgent [16] formulates video question answering (QA) as an iterative state-action-observation process, VideoTree [17] builds query-adaptive hierarchical representations, and later systems such as VCA [20], MR. Video [12], VideoDeepResearch [22], and Deep Video Discovery [25] extend agentic exploration with search, memory, and tool use. These methods show the value of adaptive evidence gathering, but they are mainly designed for offline

inference, where the full video and target question are available before reasoning. This assumption breaks in streaming scenarios, where future frames and future questions are unavailable. Our work instead studies streaming video agents that update memory online, operate within a bounded memory budget, and decide when sufficient evidence has appeared to answer.

3 Problem Formulation

Task definition. We consider a video stream $V = \{f_1, f_2, \ldots, f_T\}$, where frames arrive sequentially over time. Let $Q = \{q_1, q_2, \ldots, q_n\}$ denote the set of questions associated with the video. During inference, questions are revealed online: each question q_i becomes available only at its query time t_i during the video stream. For each question, the model must produce an answer a_i and an answer time t_i^a, indicating when it commits to the answer.

The task is subject to two constraints. First, under the *streaming constraint*, the model processes frames in temporal order and cannot revisit past frames. At time t, it can only use the current input and its internal memory state M_t. Second, under the *memory constraint*, the model maintains a bounded memory state with a fixed budget C, i.e., $\forall t, |M_t| \leq C$. Thus, the model must decide what information to retain, compress, or discard as the stream evolves.

Each question has a ground-truth answer time g_i, defined as the earliest timestep no earlier than t_i at which the question becomes answerable from the observed stream. This induces three temporal question types: **Past questions** refer to evidence that appeared before the query time; if the relevant information is preserved in memory, the question is answerable immediately, so $g_i = t_i$. **Present questions** depend on current or very recent observations and are also answerable at query time, so $g_i = t_i$. **Future questions** require evidence that has not yet appeared when the question is asked; the model must defer answering until sufficient evidence becomes available, so $g_i > t_i$. We use accuracy and answer latency to evaluate the performance on our setting, with metric definitions provided in Appendix C.1.

Key setting properties. Our formulation differs from both offline video QA and many existing streaming video settings. Compared with offline video understanding, the model cannot access the full video or dynamically inspect arbitrary temporal segments. Compared with prior streaming settings [2, 6, 9, 10, 13, 24] that often focus on short videos, single-query evaluation, or only past/current questions, our setting considers long video streams, bounded shared memory, multiple questions revealed over time, and unified past, present, and future questions without task-type labels.

A central property of our setting is its *unified temporal protocol*. Past, present, and future questions are mixed under the same streaming protocol, and the model is not given the question type during inference. Therefore, it cannot rely on oracle routing, such as always retrieving memory for past questions, answering from the current observation for present questions, or automatically deferring future questions. Instead, for each active question, the model must infer from the observed stream and its memory whether the question is answerable now or should be deferred.

Another important property is the *multi-question streaming setting*. Unlike standard video QA settings that consider a single question per video, we assume that multiple questions may arrive at different times over the same video stream. These questions remain active until answered. Since future questions are not known when earlier frames are processed, the memory cannot be optimized only for currently active questions. It must preserve question-agnostic information about the stream while also supporting question-specific evidence retrieval for active queries.

These properties introduce several key challenges: (1) **Strict sequential access:** the model must process frames in temporal order and cannot revisit past frames or inspect future frames; (2) **Bounded shared memory:** questions may depend on distant past events, but all questions must share a fixed-size memory state; (3) **Answer timing under partial observability:** for each active question, the model must decide whether the observed evidence is sufficient to answer or whether it should wait for future frames; (4) **Unified multi-question reasoning:** past, present, and future questions are handled under the same protocol, and the model must answer them without oracle task-type labels.

Overall, this setting turns long-video QA from an offline full-video inference problem into an online decision problem over memory, evidence, and answer timing.

4 Benchmark Construction

The key to constructing a streaming QA benchmark is identifying when the visual evidence for each question appears. Given such evidence timing, an offline QA example can be converted into different streaming scenarios by placing the query before, at, or after the evidence. We therefore propose Ref2Stream (*Reference-to-Stream*), a general benchmark-conversion protocol that transforms temporally grounded offline video QA examples into streaming QA instances. Ref2Stream uses reference time intervals from offline video understanding benchmarks as proxies for evidence intervals, and assigns each converted question a query time t and a ground-truth answer time g.

Ref2Stream can be applied to any offline video understanding benchmark with reference time annotations. Specifically, we assume each source example consists of a video V, a question q, answer candidates, a ground-truth answer a, and a reference time interval $[l, r]$ indicating where the supporting visual evidence appears. Let T denote the video length. The protocol preserves the original question, answer candidates, and ground-truth answer whenever possible, and generates streaming instances by changing only the temporal placement of the query.

Given a source example with reference interval $[l, r]$, Ref2Stream instantiates the three temporal question types as follows. For **past** questions, we sample $t \sim \mathcal{U}([r, T])$ and set $g = t$, since the evidence has already appeared when the question is asked and the question should be answerable immediately if the relevant information has been preserved in memory. For **present** questions, which are only constructed for point-like annotations ($l = r$), we set $t = g = l$, so the question is asked when the relevant evidence appears. For **future** questions, we sample $t \sim \mathcal{U}([0, r])$ and set $g = r$, treating the end of the reference interval as the time when the visual evidence is fully observed.

We instantiate Ref2Stream on LVBench [15], an offline long-video understanding benchmark with reference time intervals, resulting in LVBench-Ref2Stream. LVBench is well suited for this instantiation because its long videos stress long-horizon memory and its reference intervals provide the temporal evidence needed by Ref2Stream. After filtering examples that are not suitable for our streaming setting, we retain 977 source QA examples. Temporal variants derived from the same source QA share the same question, answer candidates, and ground-truth answer. Evaluating multiple such variants in one stream can introduce dependence, since answering or tracking one variant may influence another through shared memory. To avoid this potential cross-variant information leakage, we randomly select exactly one available temporal variant for each retained source QA before inference: past, present, or future for point-like reference time annotations, and past or future otherwise. The selected variants and their query times are fixed across methods and ablations. Questions are pooled by video and revealed only at their assigned query times, without temporal-type labels. Details of LVBench preprocessing, filtering rules, and temporal-variant selection are provided in Appendix C.2.

5 PIVOT: A Unified Agentic Framework for Streaming Long-Video Question Answering

Figure 2 provides an overview of PIVOT. Designed for the unified setting in Section 3, PIVOT is a modular streaming video agent that separates online video processing into two stages: *streaming memory maintenance* and *agentic question processing*. As the video stream unfolds, PIVOT incrementally updates a bounded structured memory from observed visual content; for each active question, a planner then coordinates memory retrieval, question analysis, reflection, and answer/defer decisions.

Since streaming video understanding imposes strict temporal access constraints, PIVOT must process the video in order without revisiting arbitrary past frames or accessing future frames. In practice, we use short consecutive chunks as the basic processing unit. Chunk-level processing is more suitable than frame-level processing because many visual semantics are defined over local temporal context: actions, interactions, motion cues, event boundaries, and state changes often cannot be reliably inferred from isolated frames. Thus, processing videos chunk by chunk preserves short-term scene dynamics while still respecting the streaming constraint.

Formally, let C_k denote the k-th incoming video chunk, $\mathcal{M}_k$ denote the memory state after processing C_k, and $\mathcal{Q}_k$ denote the set of active unanswered questions when processing chunk k. At each chunk, PIVOT performs two stages: (1) **Streaming Memory Maintenance**, which updates the bounded structured memory $\mathcal{M}_{k-1} \rightarrow \mathcal{M}_k$ using the current chunk C_k; and (2) **Agentic Question Processing**,

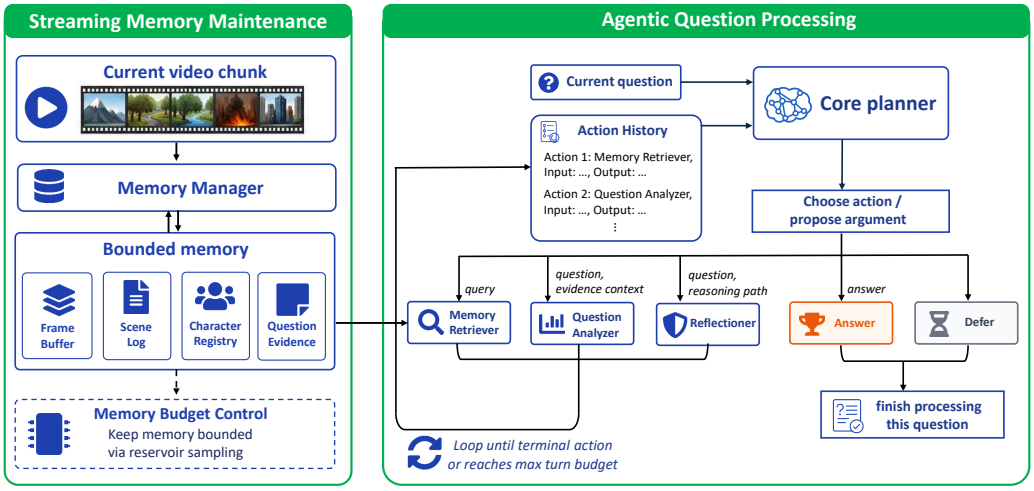

Figure 2: *Overview of PIVOT*. PIVOT processes a long video stream in short chunks and separates online inference into two stages: streaming memory maintenance and agentic question processing. For each incoming chunk, a memory manager updates a bounded structured memory consisting of a frame buffer, scene log, character registry, and question-specific evidence notes, while memory budget control keeps the retained context compact. For each active question, a core planner then runs an action loop over a small set of tools: it can retrieve relevant memory, analyze the current chunk together with retrieved evidence, reflect on candidate answers, and decide whether to answer immediately or defer the question to future chunks. This design allows PIVOT to handle past, present, and future questions under the same streaming protocol without task-type labels.

where for each $q_i \in \mathcal{Q}_k$, a core planner decides a sequence of actions to gather evidence, verify candidate answers, and either finalize an answer or defer the question.

This decomposition is important in the streaming setting. Memory maintenance must be performed before observing future frames or future questions, while question processing selectively uses the maintained memory to handle past, present, and future questions without task-type labels.

5.1 Streaming Memory Maintenance

Memory state. We maintain a bounded structured memory with four components: a transient frame buffer, a scene log, a character registry, and question-specific evidence notes.

The *frame buffer* is used to support chunk-level processing under strict streaming constraints. Since frames arrive one by one, the system must temporarily store the frames within the current chunk before summarizing them into memory. The *scene log* is an ordered list of text captions, where each entry is generated at one memory-update step. Each caption records the salient visual content observed in that update, including actions, interactions, temporal order, state changes, on-screen text, and other important visual details. The *character registry* stores persistent identity information about recurring characters. Each entry records stable visual cues, such as appearance, clothing, role, and distinguishing attributes. Its main purpose is to maintain a consistent naming space for the scene log: when a new character appears, the memory manager assigns a stable tag, and when the same character reappears, the system reuses or updates the corresponding registry entry. This helps later memory retrieval and question analysis connect events involving the same character across long videos. The *question evidence* stores compact notes for active questions. Unlike the scene log, which is shared across questions, question evidence is query-specific and records observations that may support answering this specific question.

Chunk-level memory update. As frames arrive, they are appended to the frame buffer. Once the buffer reaches the chunk size, or the video ends, the memory manager performs one memory-update step before any question-specific reasoning is performed. To reduce information loss at chunk boundaries, the visible window for chunk C_k contains the current chunk frames and, for $k > 1$, a small number of trailing frames retained from C_{k-1}. For the first chunk, no overlap is used.

Given this visible window, the memory manager produces one new scene-log entry, updates the character registry if new or recurring characters appear, and updates evidence notes for currently active questions when the visible window contains relevant information. The current chunk frames remain available during question processing for the same chunk. After processing chunk C_k, the system discards the raw frames in the buffer except for the tail frames used as overlap context for the next update. This overlap provides only local boundary context and does not allow arbitrary access to older frames.

Memory budget control. As defined in Section 3, the memory state must remain bounded throughout streaming. The transient frame buffer is naturally bounded by the chunk and overlap sizes. For other parts of the memory, PIVOT uses a token-budgeted reservoir sampling strategy to retain a diverse subset of scene-log and character-registry entries, while keeping question-evidence notes compact for active questions. Implementation details are provided in Appendix D.1.

5.2 Agentic Question Processing

After memory is updated, each active question is processed independently by a core planner. We formulate this stage as an agentic decision process: instead of directly generating an answer in a single forward pass, the planner interacts with a compact action space to gather evidence, verify candidate answers, and decide whether the question is answerable at the current time.

At each step, the planner observes the current question and the action history, including inputs and outputs from previous tool calls, and selects an action from $\mathcal{A} = \{\texttt{retrieve}, \texttt{analyze}, \texttt{reflect}, \texttt{answer}, \texttt{defer}\}$. The actions `retrieve`, `analyze`, and `reflect` are tool calls for evidence retrieval, visual-text reasoning, and answer verification, respectively. The actions `answer` and `defer` are terminal decisions: the former commits a final answer to the question and records the current time as the answer time, while the latter keeps the question active for future chunks.

This design allows the planner to adapt its reasoning path to different temporal question types. A past question may require retrieving historical evidence from memory, a present question may rely mainly on the current chunk, and a future question may need to be deferred until relevant evidence appears. Importantly, the planner is not given the question type and must infer whether to retrieve, analyze, verify, answer, or wait based on the available evidence. Detailed descriptions of the action inputs, outputs, and tool prompts are provided in Appendix D.2.

6 Experiments

We evaluate PIVOT on LVBench-Ref2Stream, our streaming long-video understanding benchmark described in Section 4. Our experiments are designed to answer three questions: (1) how existing open-source MLLMs, closed-source MLLMs, and prior streaming video methods perform under our strict streaming long-video setting; (2) whether planner-guided agentic reasoning improves answer accuracy and latency; and (3) which components of the proposed framework contribute most to the final performance. We further evaluate PIVOT on existing streaming video understanding benchmarks to study its generalization ability in Section 6.5, with full results provided in Appendix E.

6.1 Experimental Setup

Benchmark. Our main experiments are conducted on LVBench-Ref2Stream for streaming long-video understanding. Each video is processed sequentially, and questions are revealed only at their query times. We use the fixed one-variant-per-source-QA protocol described in Section 4 to avoid potential information leakage between temporal variants of the same source QA. The model does not receive task-type labels and must decide whether to answer or defer each active question based only on the observed stream and maintained memory. For the main experiments, we sample videos at 0.5 FPS and process the videos in 16 frames per chunk.

Metrics. We report answer accuracy and answer latency to evaluate answer correctness and response delay, respectively. Accuracy is our primary metric and measures multiple-choice correctness across all questions; unanswered questions are counted as incorrect. Latency measures the nonnegative delay after the annotated answerable time, in seconds on the video timeline; early responses receive

Table 1: Main results on LVBench-Ref2Stream. We report accuracy and latency overall and for each temporal question type. PIVOT achieves the best overall accuracy and latency, while FIFO and reservoir visual buffers reveal a trade-off between recent evidence and long-term retention. Higher accuracy and lower latency are better.

Method	Overall		Past		Present		Future	
	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓
Generic MLLMs with FIFO visual-context wrapper								
`Qwen2.5-VL-7B-Instruct` [14]	26.7	136.2	24.1	233.4	38.1	186.8	27.7	35.7
`LLaVA-OneVision-7B` [7]	34.2	122.9	32.3	208.2	57.1	176.9	32.9	34.1
`InternVL-8B` [3]	33.3	127.7	25.2	217.8	52.4	196.4	38.5	33.5
`Gemini-3.1-flash-lite` [5]	44.8	97.2	31.4	181.4	79.4	42.9	**53.2**	27.4
`GPT-4o` [11]	46.7	89.6	36.1	178.7	77.8	**31.3**	52.8	19.8
Generic MLLMs with reservoir visual-context wrapper								
`Gemini-3.1-flash-lite` [5]	44.1	85.3	41.4	83.6	54.0	116.4	45.5	82.4
`GPT-4o` [11]	46.0	89.8	44.5	98.4	57.1	101.6	45.9	79.7
Prior streaming video methods								
`Dispider` [13]	32.8	45.6	32.3	71.9	42.9	72.4	31.8	**11.9**
`TimeChat-Online` [21]	23.5	40.8	24.6	**62.4**	42.9	51.5	19.9	18.2
PIVOT (Ours)	**56.6**	**37.6**	**56.9**	66.0	**81.0**	36.3	53.0	12.2

zero latency. We report overall results and type-specific results for past, present, and future questions. Detailed metric definitions are provided in Appendix C.1. We additionally report Answer Readiness Score (ARS) following StreamReady [1] in Appendix C.4.

Backbone model. Unless otherwise specified, all modules of our framework use `Gemini-3.1-Flash-Lite-Preview` as the underlying multimodal reasoning model.

Memory budget. Unless otherwise specified, all methods are evaluated under a fixed 50K token-equivalent memory budget. Prior streaming memory work often reports 50K-token Full-KV settings as a reference point and evaluates compressed memories under smaller KV budgets [6, 19]. In our setting, this budget counts all retained streaming context. For PIVOT, the transient frame buffer stores only the current chunk and a short overlap window, so its cost is included but remains small and does not grow with video length. We estimate text token usage for text x with heuristic: $\mathrm{tokens}(x) \approx |x|/4$, and estimate frame cost using the measured visual-token usage per sampled frame. The 50K budget is shared by all questions in a video, not allocated per question. Retrieved context is a view of already-counted memory; temporary planner histories and tool outputs are accounted for in API usage rather than counted as retained streaming memory. Sensitivity analyses of the memory budget and chunk size are reported in Appendix G.

6.2 Baselines

We compare our method with two families of baselines: generic multimodal LLMs adapted to our streaming protocol, and prior streaming video understanding methods.

Generic MLLMs with visual-context wrappers. We adapt offline MLLMs to our streaming protocol with bounded visual-context buffers, where each model receives only retained observed frames and must either answer or defer without future-frame, future-question, or task-type access. We evaluate open-source models with FIFO buffers and closed-source/API models with FIFO and reservoir buffers under the same 50K token-equivalent budget, details are in Appendix C.3.

Prior streaming video methods. We compare with prior streaming video understanding methods that are compatible with our answer/defer evaluation, including Dispider [13] and TimeChat-Online [21]. We exclude VideoLLM-online [2] from this comparison because it mainly targets streaming dialogue and assistant-style responses rather than multiple-choice video QA.

6.3 Main Results

Table 1 compares PIVOT with generic MLLM baselines and prior streaming video methods on LVBench-Ref2Stream. It shows that PIVOT achieves the best overall performance on LVBench-

Table 2: Ablation study on the evaluation set of LVBench-Ref2Stream. Removing any component reduces overall accuracy, showing that structured memory, modular question processing, and planner-guided control all contribute. Removing the planner lowers overall latency but sacrifices accuracy, while removing modular question processing causes the largest accuracy drop.

Method	Overall		Past		Present		Future	
	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓
PIVOT	**56.6**	37.6	**56.9**	66.0	**81.0**	36.3	**53.0**	**12.2**
w/o Planner	53.0	**36.8**	54.0	**63.3**	77.8	**35.9**	48.7	13.1
w/o Modular Question Processing	48.7	44.1	53.5	72.9	79.4	40.8	39.8	16.9
w/o Structured Memory	50.2	70.9	47.6	134.7	76.2	64.4	49.1	16.6

Ref2Stream. It obtains the highest overall accuracy of 56.6 and the lowest overall latency of 37.6 seconds. Compared with the strongest direct MLLM baseline, `GPT-4o` with FIFO visual context, PIVOT improves overall accuracy by 9.9 points while reducing latency by 52.0 seconds. Compared with prior streaming video methods, PIVOT also improves both overall accuracy and overall latency. These results suggest that PIVOT is effective for streaming long-video understanding under bounded memory.

The evaluated closed-source MLLMs are substantially stronger than open-source MLLMs under the same streaming wrapper. With FIFO visual context, Gemini and `GPT-4o` achieve 44.8% and 46.7% overall accuracy, while open-source MLLMs range from 26.7% to 34.2%. This gap suggests that stronger multimodal reasoning helps in our setting, although a strong backbone alone does not recover PIVOT's performance.

The results highlight a trade-off between recent-frame retention and long-term memory. FIFO buffers preserve current visual context and therefore perform well on present and future questions, but they evict earlier evidence and suffer on past questions. Reservoir buffering improves historical coverage, increasing past accuracy from 31.4% to 41.4% for Gemini and from 36.1% to 44.5% for `GPT-4o`, but reduces present/future accuracy and increases latency on those questions. PIVOT addresses this raw-frame retention trade-off by combining bounded structured memory with agentic question processing. It stores scene descriptions, character identities, and question-specific evidence, while a planner decides whether to retrieve memory, analyze the current chunk, reflect, answer, or defer. This yields the highest past and present accuracy of 56.9% and 81.0%, respectively, while retaining competitive future accuracy of 53.0%, compared with 53.2% for Gemini FIFO. These results suggest that streaming long-video QA benefits from adaptive reasoning over both memory and current observations.

6.4 Ablation Study

We ablate the major components of our framework to understand their contributions. All ablations use the same streaming protocol, backbone model, frame sampling, and memory budget. The ablations use the same fixed one-variant-per-source-QA evaluation set as the main results.

We consider the following ablations:

(1) **w/o structured memory:** replace the structured memory with a single plain-text buffer that is truncated from the front when it exceeds the memory budget;

(2) **w/o modular question processing:** keeps the structured memory, but collapses memory update and question handling into a single chunk-level model call. The model receives the previous memory, the current chunk, and all active questions, and directly outputs the updated memory together with answer/defer decisions;

(3) **w/o planner:** keep the modular architecture, but replace planner-guided control with a fixed workflow that retrieves memory, analyzes the question, and then answers or defers, without reflection;

Analysis. Table 2 shows that all three components contribute to streaming long-video QA. Removing any component reduces accuracy, with the largest overall drop of 7.9 percentage points caused by removing modular question processing. The drop is particularly large for future questions, whose accuracy falls from 53.0% to 39.8%. This suggests that separating streaming memory maintenance from agentic question processing is important for repeatedly assessing whether the necessary evidence has appeared. Replacing structured memory with a plain-text buffer reduces overall accuracy from

56.6% to 50.2% and increases latency from 37.6 to 70.9 seconds. The effect is especially pronounced for past questions: accuracy drops from 56.9% to 47.6%, while latency increases from 66.0 to 134.7 seconds. These results indicate that the scene log, character registry, and question-evidence notes help preserve and retrieve long-range evidence.

The **w/o planner** variant has slightly lower overall latency than the full model (36.8 versus 37.6 seconds), but this comes at the cost of 3.6 accuracy points (53.0% versus 56.6%). This comparison evaluates the planner-controlled reasoning loop, including adaptive action selection and reflection, as a whole rather than isolating adaptive planning alone. Overall, the ablation results show that modular question processin g provides the largest gain, structured memory improves long-range evidence retention, and the full planner-controlled reasoning loop further improves accuracy over the fixed workflow.

6.5 Evaluation on Existing Streaming Benchmarks

We further evaluate PIVOT on two existing streaming video benchmarks, StreamingBench [9] and OVO-Bench [10]. These benchmarks use different task definitions and evaluation protocols from LVBench-Ref2Stream, so we treat them as additional generalization studies. For PIVOT, we follow a streaming inference protocol: the model processes video inputs sequentially, maintains bounded memory, and does not access future frames or future questions during inference.

Table 3: Overall scores on existing streaming benchmarks for PIVOT and prior online/streaming methods. Higher is better. "–" denotes an unreported overall score.

Method	StreamingBench ↑	OVO-Bench ↑
Flash-VStream	24.0	33.6
VideoLLM-online	32.5	–
Dispider	53.1	41.8
TimeChat-Online	58.0	46.7
PIVOT	**66.2**	**61.9**

Table 3 summarizes the overall results. PIVOT achieves the highest overall scores among the compared online/streaming video methods: 66.2 on StreamingBench and 61.9 on OVO-Bench, exceeding TimeChat-Online by 8.2 and 15.2 points, respectively. These results suggest that the proposed framework generalizes beyond LVBench-Ref2Stream to existing streaming video benchmarks. Full task-level results and comparisons with offline reference models are provided in Appendix E.

6.6 Compute Resources

All experiments are inference-only, with no training or fine-tuning. We profile the complete PIVOT pipeline, including streaming memory maintenance and all question-processing calls. Under the profiled configuration, each video requires an average of 491.2 API calls at an API cost of $2.26. The mean ratio of video duration to wall-clock processing time is 3.19×, computed per video and then averaged. For 32-second chunks, the median and 90th-percentile wall-clock processing times are 7.39 and 19.95 seconds, respectively, and 96.7% of chunks finish within 32 seconds. These measurements support average-throughput real-time feasibility. And detailed API usage and timing statistics are provided in Appendix F.

7 Conclusion

We introduce PIVOT, a unified agentic framework for streaming long-video understanding under strict sequential access and bounded shared memory. We also propose Ref2Stream and instantiate it on LVBench to construct LVBench-Ref2Stream, which evaluates past, present, and future questions with both answer accuracy and latency. Experiments show that structured memory, modular question processing, and planner-guided control all contribute to improved streaming long-video understanding, suggesting that this problem should be viewed not only as memory compression, but also as online agentic decision-making over evolving evidence.

Acknowledgments and Disclosures

Funding and Computational Resources. This work was supported in part by the Apple AIML Academic Research Program, NSF Grants 2106825 and 2519216, the DARPA Young Faculty Award, ONR Grant N00014-26-1-2099, and NIFA Award 2020-67021-32799. This work used computational resources, including API credits from the Google Gemini Academic Program, the NCSA Delta and DeltaAI supercomputers under allocations CIS230012, CIS230013, CIS240133, and CIS240387 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, and Amazon Web Services (AWS), Microsoft Azure, and the OpenAI API through the National Artificial Intelligence Research Resource (NAIRR) Pilot.

Competing Interests. The authors declare no additional competing financial interests beyond the affiliations disclosed in this paper.

References

[1] Shehreen Azad, Vibhav Vineet, and Yogesh S Rawat. StreamReady: Learning what to answer and when in long streaming videos. In *CVPR*, 2026. 3, 8, 15

[2] Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. VideoLLM-online: Online video large language model for streaming video. In *CVPR*, 2024. 1, 3, 4, 8

[3] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yiming Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024. 8

[4] 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 *ICLR*, 2025. 3

[5] Google AI for Developers. Gemini 3.1 Flash-Lite Preview. Google AI for Developers model documentation, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.1-flash-lite-preview. Accessed: 2026-05-07. 8

[6] Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. InfiniPot-V: Memory-constrained KV cache compression for streaming video understanding. In *NeurIPS*, 2025. 3, 4, 8

[7] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *TMLR*, 2025. 8

[8] Zhijia Liang, Jiaming Li, Weikai Chen, Yanhao Zhang, Haonan Lu, and Guanbin Li. OASIS: On-demand hierarchical event memory for streaming video reasoning. In *CVPR*, 2026. 16

[9] Junming Lin, Zheng Fang, Chi Chen, Haoxuan Cheng, Zihao Wan, Fuwen Luo, Ziyue Wang, Peng Li, Yang Liu, and Maosong Sun. StreamingBench: Assessing the gap for MLLMs to achieve streaming video understanding. In *ICASSP*, 2026. 1, 2, 3, 4, 10

[10] 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 *CVPR*, 2025. 1, 2, 3, 4, 10

[11] OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 8

[12] Ziqi Pang and Yu-Xiong Wang. MR. Video: MapReduce as an effective principle for long video understanding. In *NeurIPS*, 2025. 3

[13] 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 *CVPR*, 2025. 3, 4, 8

[14] Qwen Team. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025. 8

[15] 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 *ICCV*, 2025. 1, 3, 5, 14, 16

[16] Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. VideoAgent: Long-form video understanding with large language model as agent. In *ECCV*, 2024. 1, 3

[17] 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 *CVPR*, 2025. 1, 3

[18] Jiaer Xia, Peixian Chen, Mengdan Zhang, Xing Sun, and Kaiyang Zhou. Streaming video instruction tuning. In *CVPR*, 2026. 3

[19] Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. StreamMem: Query-agnostic KV cache memory for streaming video understanding. In *CVPR Workshop on Video LLMs*, 2026. 3, 8

[20] Zeyuan Yang, Delin Chen, Xueyang Yu, Maohao Shen, and Chuang Gan. VCA: Video curious agent for long video understanding. In *ICCV*, 2025. 3

[21] 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 *ACM MM*, 2025. 3, 8

[22] Huaying Yuan, Zheng Liu, Junjie Zhou, Ji-Rong Wen, and Zhicheng Dou. VideoDeepResearch: Long video understanding with agentic tool using. *arXiv preprint arXiv:2506.10821*, 2025. 3

[23] 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 *NeurIPS*, 2025. 3

[24] Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-VStream: Efficient real-time understanding for long video streams. In *ICCV*, 2025. 3, 4

[25] 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 *NeurIPS*, 2025. 1, 3

A Limitations and Future Work

This work has several limitations. First, our computational profiling supports average-throughput real-time feasibility, but does not establish a hard per-chunk latency guarantee. Second, PIVOT processes videos chunk by chunk. While chunk-level processing preserves local temporal context for actions, interactions, and motion cues, it also introduces chunk-induced latency because memory update and question processing are performed only after a chunk has been accumulated. Third, the memory in PIVOT is primarily text-based. Although textual memory is compact and easy to retrieve, it may discard fine-grained visual details. Future work could incorporate visual memory, such as selected keyframes or visual embeddings, to better preserve important visual evidence over long streams.

B Broader Impacts

This work studies streaming long-video understanding, where models process visual inputs over time and answer questions when sufficient evidence has been observed. The potential positive impact is that such systems could support useful real-world applications, including assistive video agents, long-video search, robotics, augmented-reality assistants, and monitoring systems that help users understand ongoing events without requiring full offline video review. By emphasizing bounded memory, answer timing, and deferral when evidence is insufficient, our formulation also encourages streaming video systems to reason more carefully about when they should answer rather than always responding immediately.

At the same time, streaming video understanding also raises important societal risks. Since visual inputs may arrive continuously, systems built for this setting could be misused for surveillance, tracking, or profiling people without appropriate consent. Even when used for benign purposes, models may store or summarize sensitive visual information, including personal appearance, activities, locations, or interactions. Incorrect answers or premature responses may also cause harm if such systems are used in safety-critical or high-stakes settings, such as security monitoring, healthcare, autonomous systems, or workplace decision making. In addition, performance may vary across environments, visual conditions, cultures, and demographic groups, potentially leading to uneven reliability.

Our work is intended as a research benchmark and method for studying streaming long-video understanding, not as a ready-to-deploy surveillance or decision-making system. Deployments of this type of technology should include strong safeguards, such as consent and notice for recorded subjects, strict access control, privacy-preserving memory policies, data minimization, limits on retention of raw video or identity information, human oversight for high-stakes decisions, and auditing for bias and failure cases. Future work should further study privacy-preserving streaming memory, safer answer/deferral calibration, and evaluation protocols that explicitly measure fairness, privacy leakage, and misuse risks.

C Additional Details

C.1 Metric Definitions

We evaluate models with answer accuracy and answer latency.

Answer accuracy. Answer accuracy measures the correctness of the predicted multiple-choice answer across all questions. Unanswered questions are counted as incorrect.

Answer latency. Answer latency measures the delay between when a question becomes answerable and when the model commits to an answer.

Let $\mathcal{I}_{\text{ans}}$ denote the set of indices of questions answered by the model, regardless of answer correctness. We compute

$$\text{Latency} = \frac{1}{|\mathcal{I}_{\text{ans}}|} \sum_{i \in \mathcal{I}_{\text{ans}}} \max(0, t_i^a - g_i),$$

where g_i is the annotated answerable time and t_i^a is the model's answer time, both measured in seconds on the video timeline. Type-specific latency is computed over the answered questions of the corresponding temporal type.

We only penalize *late responses*. This design accounts for the fact that ground-truth answer time annotations may be imprecise in practice: an early correct answer can indicate that sufficient evidence appears earlier than the annotated reference time. Therefore, early answers receive zero latency penalty, while answer correctness is still evaluated by answer accuracy. Unanswered questions are excluded from both the latency sum and its denominator, but are counted as incorrect for answer accuracy. We additionally report Answer Readiness Score (ARS) in Appendix C.4.

C.2 LVBench-Ref2Stream Details

We instantiate Ref2Stream on LVBench [15], an offline long-video understanding benchmark with reference time annotations. LVBench contains 103 videos and 1,549 question-answer pairs, with an average video duration of 4,101 seconds. Its long video duration makes it suitable for evaluating long-horizon memory under streaming constraints, while its reference time intervals provide the temporal evidence needed to construct query-time and answer-time annotated streaming QA instances.

Before applying Ref2Stream, we filter LVBench examples that are not suitable for our streaming setting. First, we remove questions that explicitly require absolute timestamp prediction or contain absolute timestamps in the question. In our streaming setting, the model should reason from the current stream state rather than rely on offline access to the complete video timeline. Second, we remove summarization questions, since they often require holistic access to a large portion of the video and do not naturally correspond to a localized answer time. For the same reason, we also remove examples whose reference interval covers more than 80% of the full video duration.

After filtering, we retain 977 source QA examples. For each retained source QA, we randomly select exactly one available temporal variant before inference: past, present, or future for point-like reference intervals, and past or future otherwise. The selected variants and their query times are fixed across methods and ablations. During evaluation, questions are pooled by video and revealed only at their assigned query times. The model receives no task-type labels, and unanswered questions remain active until the model commits an answer or the video ends. Thus, each source QA appears only once in a video stream, while multiple distinct questions share the same bounded memory.

C.3 Generic MLLM Streaming Adaptation

Most general-purpose multimodal LLMs are designed for offline image or video understanding and do not natively maintain persistent streaming memory. We therefore adapt them to our setting with simple visual-context wrappers. At each chunk, the wrapper maintains a fixed-size buffer of observed frames under the same 50K token-equivalent memory budget. For each active question, the model receives the buffered frames, the question, and the answer candidates, and must output either an answer option or defer. If the model defers, the question remains active and is queried again at later chunks. This wrapper ensures that the model never accesses future frames, future questions, or task-type labels.

We consider two buffer update policies. The default policy is a *FIFO* buffer, which keeps the most recently observed frames and therefore emphasizes local visual context. We also evaluate a *reservoir* buffer for selected closed-source models, where frames are maintained with standard reservoir sampling over the observed prefix. This provides a simple streaming baseline with more uniform long-range coverage of past frames, while still satisfying the same bounded-memory and no-future-access constraints.

Using these streaming adaptations, we evaluate both open-source and closed-source/API multimodal models. For open-source models, we consider representative video/image MLLMs including `Qwen2.5-VL-7B-Instruct`, `LLaVA-OneVision-7B`, and `InternVL-8B` with the FIFO wrapper. For closed-source/API models, we evaluate `Gemini-3.1-Flash-Lite` and `GPT-4o`, and report both FIFO and reservoir variants. In particular, the direct `Gemini-3.1-Flash-Lite` baseline uses the same backbone as PIVOT, allowing us to isolate the benefit of our streaming memory and agentic design from the capability of the underlying model.

C.4 Answer Readiness Score (ARS)

Following the ARS formulation of StreamReady [1], we evaluate response timing using early and late penalties. For question i, let $t_{a,i} = t_i^a$ denote the answer time, $t_{s,i} = l_i$ the start of its source reference interval, and $t_{e,i} = g_i$ its annotated answerable time. Thus, $t_{e,i}$ is the query time for past/present questions and the reference-interval end for future questions.

For an answered question, we compute

$$\mathrm{EP}_i = \min\left(1,\, 2\sigma\left(\gamma_e \frac{t_{a,i} - t_{s,i}}{\tau + \epsilon}\right)\right),$$

$$\mathrm{LP}_i = \min\left(1,\, \max\left(0,\, 1 - \gamma_\ell \frac{t_{a,i} - t_{e,i}}{\tau + \epsilon}\right)\right),$$

$$\mathrm{ARS}_i = \mathrm{EP}_i\, \mathrm{LP}_i,$$

where σ is the logistic sigmoid and $\tau = \mathrm{median}_{i=1,\ldots,N}(t_{e,i} - t_{s,i})$ is computed once over all N evaluation questions, pooling all temporal types. We use $\gamma_e = 0.25$, $\gamma_\ell = 0.5$, and $\epsilon = 1$ with timestamps measured in seconds. The reduced decay coefficients are chosen to accommodate the temporal granularity of the inherited LVBench annotations. The same normalization scale and parameters are used across methods and temporal-type breakdowns. Unanswered questions receive $\mathrm{ARS}_i = 0$, and the reported score is

$$\mathrm{ARS} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{ARS}_i.$$

Higher ARS is better. We report ARS on the same fixed one-variant-per-source-QA evaluation set used in Section 6, with answer accuracy repeated for reference.

Main comparison. Table 4 reports ARS for the methods in Table 1, with accuracy repeated for reference. Among the compared methods, PIVOT achieves the highest accuracy (56.6%) and ARS (0.574). Compared with `GPT-4o` with FIFO visual context, the strongest baseline on both metrics, PIVOT improves accuracy by 9.9 percentage points and ARS by 0.090. Compared with the same-backbone Gemini FIFO baseline, PIVOT improves accuracy by 11.8 percentage points and ARS by 0.113.

Table 4: ARS results on LVBench-Ref2Stream under the fixed one-variant-per-source-QA protocol. Accuracy is repeated from Table 1 for reference. Higher accuracy and ARS are better.

Method	Acc. (%) ↑	ARS ↑
Generic MLLMs with FIFO visual-context wrapper		
`Qwen2.5-VL-7B-Instruct`	26.7	0.128
`LLaVA-OneVision-7B`	34.2	0.323
`InternVL-8B`	33.3	0.247
`Gemini-3.1-flash-lite`	44.8	0.461
`GPT-4o`	46.7	0.484
Generic MLLMs with reservoir visual-context wrapper		
`Gemini-3.1-flash-lite`	44.1	0.472
`GPT-4o`	46.0	0.471
Prior streaming video methods		
`Dispider`	32.8	0.318
`TimeChat-Online`	23.5	0.447
PIVOT	**56.6**	**0.574**

Component ablations. Table 5 reports ARS for the component ablations in Table 2, with accuracy repeated for reference. Removing structured memory reduces overall ARS from 0.574 to 0.466, with a particularly large decrease on past questions (0.633 to 0.424). Removing modular question processing reduces overall ARS to 0.542. The variant without the planner has slightly higher overall ARS (0.579 versus 0.574), but lower accuracy (53.0% versus 56.6%). Thus, the full model improves accuracy by 3.6 percentage points with only a 0.005 lower overall ARS. On future questions, however, the full model achieves higher ARS than the variant without the planner (0.490 versus 0.474).

Table 5: ARS results for the component ablations in Table 2. Accuracy (percent) is repeated for reference. All configurations use the same fixed one-variant-per-source-QA evaluation set. Higher values are better.

Method	Overall		Past		Present		Future	
	Acc. ↑	ARS ↑	Acc. ↑	ARS ↑	Acc. ↑	ARS ↑	Acc. ↑	ARS ↑
PIVOT	**56.6**	0.574	**56.9**	0.633	**81.0**	0.774	**53.0**	**0.490**
w/o Planner	53.0	**0.579**	54.0	**0.659**	77.8	**0.781**	48.7	0.474
w/o Modular Question Processing	48.7	0.542	53.5	0.619	79.4	0.763	39.8	0.436
w/o Structured Memory	50.2	0.466	47.6	0.424	76.2	0.775	49.1	0.466

C.5 Temporal Annotations and Early Responses

Ref2Stream derives its timing annotations from LVBench [15], where a reference interval identifies evidence supporting an offline QA pair. Such annotations need not enumerate every valid occurrence of the supporting evidence. When relevant evidence appears multiple times but only a later occurrence is annotated, the inherited ready time can be later than when the question first becomes answerable from the observed stream. An answer based on earlier valid evidence can therefore be classified as "early" relative to the annotation.

To assess the inherited annotations, we manually audited 50 randomly sampled instances against their videos, and found that 46/50 (92%) were correct. All four observed discrepancies were traceable to the source LVBench annotations: two reference intervals were directly incorrect in the original LVBench benchmark, while the other two marked only a later occurrence of repeated evidence. The latter cases are not necessarily errors for offline QA, where one sufficient evidence interval may be adequate, but it is non-exhaustive for streaming conversion because Ref2Stream must estimate the earliest answerable time. If all valid occurrences had been annotated, the earlier occurrence would define the ready time.

To complement this annotation-level analysis, we further examine how often PIVOT answers before the annotated ready time and how accurate these responses are. Table 6 reports the Future question timing breakdown. The early group is 41.5 accuracy points less accurate than the group answering at or after the annotated ready time, exposing the risk of premature commitment. Correct early responses may also be possible when a source interval omits an earlier valid occurrence.

Table 6: PIVOT's response timing on Future questions. "Early" is defined relative to the annotated ready time inherited from LVBench.Shares are computed over all Future questions, whereas accuracy is computed within each answered group.

Answer timing	Share (%)	Accuracy (%)
Before annotated ready time	67.1	40.6
At or after annotated ready time	31.4	82.1
Not answered	1.5	–

C.6 Contributions of Memory and Agentic Processing

PIVOT combines structured streaming memory with agentic question processing. Comparing the full system with simple FIFO or reservoir frame buffers changes both how evidence is retained and how it is used, and therefore does not distinguish their contributions to performance. Beyond the component ablations in Table 2, we study whether stronger memory alone can account for PIVOT's gains, whether agentic processing remains beneficial with stronger memory, and whether the two can be combined for further improvement. We therefore additionally follow OASIS [8], a recent strong hierarchical event-memory framework for streaming video reasoning, and construct an OASIS-style non-agentic memory baseline and a PIVOT + OASIS Memory variant.

OASIS Memory Baseline. The memory comprises a high-fidelity short window, a lower-resolution medium-term buffer, and a bounded multi-resolution Event Forest. Each incoming chunk produces an event node containing its temporal interval, text summary, semantic embedding, and sampled

keyframes. Semantically coherent adjacent nodes are greedily merged into higher-level events. Text summaries, stored keyframes, and frame buffers are included in the bounded-memory setting. At each chunk, the raw question is embedded to retrieve relevant, nonredundant event nodes. Retrieval is performed exactly once, followed by one fixed answer/defer model call for each active question. This control uses neither PIVOT's planner nor OASIS's model-controlled two-phase retrieval/refinement gate, allowing us to examine stronger memory without a planner-controlled tool loop. It is an adaptation of OASIS's memory design, not an evaluation of its complete reasoning system.

PIVOT + OASIS Memory. We replace PIVOT's text-based structured memory with this hierarchical visual-semantic memory while retaining its modular per-question processing and planner-guided tool loop. Comparing this variant with the OASIS Memory Baseline examines agentic processing with the same memory design; comparing it with PIVOT examines the benefit of stronger memory while retaining the question-processing architecture.

Evaluation setup. We evaluate all five configurations on a common subset containing 25% of the videos in LVBench-Ref2Stream, using the same Gemini backbone, 50K token-equivalent memory budget, and fixed one-variant-per-source-QA protocol. All configurations use identical videos, selected temporal variants, and query times. Table 7 reports overall accuracy on this subset.

Table 7: Memory and agentic-processing comparisons on a common 25% video subset of LVBench-Ref2Stream. Higher accuracy is better.

Method	Acc. (%) ↑
FIFO Baseline	44.7
PIVOT w/o Modular Question Processing	49.9
OASIS Memory Baseline	53.5
PIVOT	59.9
PIVOT + OASIS Memory	**61.8**

The results in Table 7 support three complementary conclusions:

Memory quality matters. Among the non-agentic controls, accuracy increases from FIFO (44.7%) to PIVOT w/o Modular Question Processing (49.9%) and the OASIS Memory Baseline (53.5%). These results support the importance of memory design even without planner-guided processing and provide a stronger memory-centered reference than a simple frame buffer.

Agentic processing contributes beyond memory quality. With PIVOT's agentic question processing, the full system improves over PIVOT w/o Modular Question Processing by 10.0 accuracy points (59.9% vs. 49.9%). With the OASIS-style memory design, PIVOT + OASIS Memory improves over the fixed-workflow OASIS Memory Baseline by 8.3 points (61.8% vs. 53.5%). PIVOT also outperforms the OASIS Memory Baseline by 6.4 points despite using its lighter text-based memory. Together, these comparisons support the contribution of modular per-question processing and adaptive control beyond memory quality alone.

Memory and agentic processing are complementary. Retaining PIVOT's question-processing architecture while replacing its memory with the OASIS-style backend improves accuracy from 59.9% to 61.8%. Thus, PIVOT is not tied to one memory construction: stronger memory enhances its performance rather than making agentic processing unnecessary.

D PIVOT Implementation Details

The overall PIVOT implementation is summarized in Algorithm 1.

D.1 Memory Budget Control

The 50K token-equivalent budget covers the full retained memory state. The frame buffer stores only the current chunk and a short overlap window from the previous chunk. Its token cost is bounded by

the fixed chunk size and overlap size, and it remains small compared to other parts of the memory and it does not grow with video length.

We enforce the remaining budget over the scene log, character registry, and question-specific evidence notes. Token usage for text is estimated by the heuristic $\text{tokens}(x) \approx |x|/4$. When a new scene-log entry is generated, it is first appended to memory. If the memory remains within the budget, the entry is kept. Otherwise, the system applies reservoir sampling: the new entry is accepted with probability K/N, where N is the number of scene entries observed so far and K is the number of scene entries currently retained. If accepted, it replaces a randomly selected retained scene entry; otherwise, it is discarded.

New character-registry entries are handled similarly, while existing character entries are updated in place when new stable identity information appears. Question-evidence notes are kept compact for active unanswered questions. As a final safeguard, if the memory still exceeds the budget, the system trims entries by randomly removing scene-log entries, then character-registry entries, and finally shortening long question-evidence notes when necessary.

D.2 Agentic Action Details

Notation. We reuse the notation from Section 5. Let C_k denote the k-th video chunk and $\mathcal{M}_k$ denote the memory state after processing C_k. Let τ_k denote the end time of chunk C_k. For an active question q_i, we use s to index the planner step within the current chunk. Let $\mathcal{H}_i^s$ denote the action history for question q_i after step s, with $\mathcal{H}_i^0 = \emptyset$. At each step, the planner returns an action-argument pair (α_i^s, η_i^s), where α_i^s is the selected action and η_i^s contains action-specific arguments.

When retrieval is applied to $\mathcal{M}_k$, it operates over the scene log, character registry, and question-evidence notes. The transient frame buffer is used only for current-chunk visual analysis and reflection, and is not treated as a retrievable long-term memory.

Memory retrieval. The `retrieve` action recovers relevant historical evidence from the memory. Given a planner-generated query u_i^s, the retriever selects useful entries from the scene log, character registry, and question-evidence notes:

$$R_i^s = \text{Retrieve}(u_i^s, \mathcal{M}_k).$$

The retrieved result R_i^s may contain selected scene descriptions, character information, and accumulated question evidence. In our implementation, retrieval is primarily performed by LLM-based selection over maintained memory entries. The retrieved result is appended to the planner's action history, allowing the planner to decide how to use it in later reasoning steps.

Question analysis. The `analyze` action invokes a visual-text reasoning tool. The planner provides a question or sub-question v_i^s and an evidence context E_i^s, which is constructed from previous tool outputs, typically including retrieved memory. The analyzer reasons over the current chunk and the provided evidence context:

$$z_i^s = \text{Analyze}(v_i^s, C_k, E_i^s).$$

The analyzer outputs either a proposed answer or a `need_more_frames` decision when the current evidence is insufficient. When an answer is proposed, it also returns a confidence score and a short reasoning summary. When more evidence is needed, it leaves the answer empty and explains what evidence is still missing.

Reflection. The `reflect` action verifies a candidate answer before finalization. The reflection agent receives the question, the proposed answer and its reasoning contained in η_i^s, the previous action history, and the current chunk frames:

$$\rho_i^s = \text{Reflect}(q_i, \eta_i^s, \mathcal{H}_i^{s-1}, C_k).$$

The reflection agent checks whether the answer is logically sound and supported by the retrieved textual evidence and current visual observations. It returns an accept/reject recommendation, a confidence score, and a short critique. If the proposed answer is accepted, the planner may finalize the answer. If it is rejected, the planner can gather additional evidence, call the analyzer again with improved context, or defer the question.

Algorithm 1 PIVOT Implementation

Require: Video stream V, questions $\{(q_i, t_i)\}_{i=1}^{n}$, chunk size L, memory budget C, maximum planner steps S

Initialize memory $\mathcal{M}_0$, frame buffer $\mathcal{B} \leftarrow \emptyset$, overlap buffer $\mathcal{O} \leftarrow \emptyset$
Initialize active question set $\mathcal{Q}_{\text{act}} \leftarrow \emptyset$
for each incoming frame f_t in temporal order **do**
  Append f_t to frame buffer $\mathcal{B}$
  if $|\mathcal{B}| = L$ or the video ends **then**
    Form current chunk $C_k \leftarrow \mathcal{B}$ and visible window $W_k \leftarrow \mathcal{O} \cup C_k$
    Reveal all questions with query time $t_i \leq \tau_k$ and add unanswered ones to $\mathcal{Q}_{\text{act}}$
    Update memory: $\mathcal{M}_k \leftarrow \text{UpdateMemory}(\mathcal{M}_{k-1}, W_k, \mathcal{Q}_{\text{act}})$
    for each active question $q_i \in \mathcal{Q}_{\text{act}}$ **do**
      Initialize action history $\mathcal{H}_i^0 \leftarrow \emptyset$
      for $s = 1$ to S **do**
        $(\alpha_i^s, \eta_i^s) \leftarrow \text{Planner}(q_i, \mathcal{H}_i^{s-1})$
        if $\alpha_i^s =$ `retrieve` **then**
          Extract retrieval query $u_i^s \leftarrow \eta_i^s$.`query`
          $R_i^s \leftarrow \text{Retrieve}(u_i^s, \mathcal{M}_k)$
          $\mathcal{H}_i^s \leftarrow \mathcal{H}_i^{s-1} \cup \{(\alpha_i^s, \eta_i^s, R_i^s)\}$
        else if $\alpha_i^s =$ `analyze` **then**
          Extract analysis query $v_i^s \leftarrow \eta_i^s$.`question` and context $E_i^s \leftarrow \eta_i^s$.`context`
          $z_i^s \leftarrow \text{Analyze}(v_i^s, C_k, E_i^s)$
          $\mathcal{H}_i^s \leftarrow \mathcal{H}_i^{s-1} \cup \{(\alpha_i^s, \eta_i^s, z_i^s)\}$
        else if $\alpha_i^s =$ `reflect` **then**
          Extract candidate answer and reasoning from η_i^s
          $\rho_i^s \leftarrow \text{Reflect}(q_i, \eta_i^s, \mathcal{H}_i^{s-1}, C_k)$
          $\mathcal{H}_i^s \leftarrow \mathcal{H}_i^{s-1} \cup \{(\alpha_i^s, \eta_i^s, \rho_i^s)\}$
        else if $\alpha_i^s =$ `answer` **then**
          Extract final answer $\hat{a}_i \leftarrow \eta_i^s$.`answer`
          Commit $\hat{a}_i$ and set answer time $t_i^a \leftarrow \tau_k$
          Remove q_i from $\mathcal{Q}_{\text{act}}$
          break
        else if $\alpha_i^s =$ `defer` **then**
          Keep q_i active for future chunks
          break
        end if
      end for
    end for
    Keep up to the last 8 frames of C_k as overlap $\mathcal{O}$; clear $\mathcal{B}$
  end if
end for

Answer and defer. The `answer` action commits a final answer and records the current time as the answer time t_i^a. The `defer` action keeps the question active when the collected evidence is insufficient. Deferred questions are reconsidered when future chunks arrive.

D.3 Prompt Details

We report the exact system prompts used by PIVOT in this section. The prompts are organized by module: memory manager, memory retriever, question analyzer, reflection agent, and core planner.

Table 8: Results on StreamingBench. Under the comparable streaming setting, PIVOT achieves the best overall performance among prior online/streaming video methods. Despite operating under a stricter streaming protocol, PIVOT also remains competitive with most offline proprietary and open-source VideoLLMs.

	Real-Time Visual Understanding (RTVU)											Omni-Source (OSU)					Contextual (CU)					
Model	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	*All*	ER	SCU	SD	MA	*All*	ACU	MCU	SQA	PO	*All*	**Overall**
Human Agents	89.5	92.0	93.6	91.5	95.7	92.5	88.0	88.8	89.7	91.3	91.5	88.0	88.2	93.6	90.3	90.3	88.8	90.4	95.0	100.0	93.5	91.7
Proprietary MLLMs																						
Gemini 1.5 pro	79.0	80.5	83.5	79.7	**80.0**	84.7	77.8	64.2	**72.0**	48.7	75.7	46.8	39.6	**74.9**	**80.0**	**60.2**	51.4	40.7	**54.8**	45.1	48.7	**67.1**
GPT-4o	77.1	80.5	83.9	76.5	70.2	83.8	66.7	62.2	69.1	49.2	73.3	41.2	37.2	43.6	56.0	44.5	41.2	38.4	32.8	56.9	38.7	60.1
Claude 3.5 Sonnet	80.5	77.3	82.0	81.7	72.3	75.4	61.1	61.8	69.3	43.1	72.4	31.6	34.0	32.8	48.8	36.8	38.4	34.8	34.4	**64.7**	37.7	57.7
Open-source Offline VideoLLMs																						
LLaVA-OneVision	80.4	74.2	76.0	80.7	72.7	71.7	67.6	65.5	65.7	45.1	71.1	40.8	37.2	33.6	44.8	38.4	35.6	36.0	27.3	29.6	32.7	56.4
Qwen2-VL-7B	75.2	82.8	73.2	77.5	68.3	71.0	72.2	61.2	61.5	46.1	69.0	41.2	22.0	32.8	43.6	34.9	31.2	26.0	39.6	22.7	31.7	54.1
MiniCPM-V 2.6	71.9	71.1	77.9	75.8	64.6	65.7	70.4	56.1	62.3	53.4	67.4	40.8	24.0	34.0	41.2	35.0	34.0	31.6	41.9	22.2	35.0	53.9
LLaVA-NeXT-Video-32B	78.2	70.3	73.8	76.8	63.4	69.8	57.4	56.1	64.3	38.9	67.0	37.7	24.8	34.4	42.8	34.9	29.2	30.4	35.4	18.2	30.8	52.8
InternVL-V2	68.1	60.9	69.4	77.1	67.7	62.9	59.3	53.2	55.0	56.5	63.7	37.6	26.4	37.2	42.0	35.8	32.0	31.2	32.3	40.9	32.4	51.4
Kangaroo	71.1	84.4	70.7	73.2	67.1	61.7	56.5	55.7	62.0	38.9	64.6	37.6	31.2	28.8	39.2	34.2	32.8	26.4	33.8	16.0	30.1	51.1
LongVA	70.0	63.3	61.2	70.9	62.7	59.5	61.1	53.7	54.7	34.7	60.0	39.6	32.4	28.0	41.6	35.4	32.8	29.6	30.3	15.9	29.9	48.7
VILA-1.5	53.7	49.2	71.0	56.9	53.4	53.9	54.6	48.8	50.1	17.6	52.3	41.6	26.4	28.4	36.0	33.1	26.8	34.0	23.2	17.6	27.4	43.2
Video-CCAM-14B	56.4	57.8	65.3	62.8	64.6	51.4	42.6	48.0	49.6	31.6	54.0	33.6	22.0	28.4	34.8	29.7	27.6	24.4	16.7	22.7	22.9	42.5
Video-LLaMA2	55.9	55.5	57.4	58.2	52.8	43.6	39.8	42.7	45.6	35.2	49.5	30.4	32.4	30.4	36.0	32.4	24.8	26.8	18.7	0.0	21.9	40.4
Open-source Online (streaming) VideoLLMs																						
Flash-VStream	25.9	43.6	24.9	23.9	27.3	13.1	18.5	25.2	23.9	48.7	23.2	25.9	24.9	25.6	28.4	26.0	24.8	25.2	26.8	2.0	24.1	24.0
VideoLLM-online	39.1	40.1	34.5	31.1	46.0	32.4	31.5	34.2	42.5	27.9	36.0	31.2	26.5	24.1	32.0	28.4	24.2	29.2	30.8	3.9	26.6	32.5
Dispider	74.9	75.5	74.1	73.1	74.4	59.9	76.1	62.9	62.2	45.8	67.6	35.5	25.3	38.6	43.3	35.7	39.6	27.6	34.8	25.3	33.6	53.1
TimeChat-Online	80.8	79.7	80.8	83.3	74.8	78.8	**78.7**	64.2	68.8	**58.0**	75.3	41.6	29.6	34.8	52.0	39.5	41.2	30.4	42.8	18.8	33.3	58.0
PIVOT	**83.5**	**84.4**	**85.8**	**85.3**	74.2	**90.7**	62.7	**74.4**	63.4	**58.0**	**77.3**	**47.2**	**46.8**	51.2	69.2	53.6	**59.6**	**53.2**	44.4	48.5	**51.4**	66.2

Table 9: Results on OVO-Bench. PIVOT achieves the best overall performance among online/streaming video methods and substantially improves backward tracing performance, showing the benefit of planner-guided memory retrieval for long-horizon streaming reasoning.

		Real-Time Visual Perception							Backward Tracing				Forward Active Responding				
Model	**#F**	OCR	ACR	ATR	STU	FPD	OJR	*Avg*	EPM	ASI	HLD	*Avg*	REC	SSR	CRR	*Avg*	**Overall**
Human Agents	–	94.0	92.6	94.8	92.7	91.1	94.0	93.2	92.6	93.0	91.4	92.3	95.5	89.7	93.6	92.9	92.8
Blind LLMs																	
GPT-4-turbo	–	28.9	24.8	25.7	33.8	27.7	26.6	27.9	42.8	48.6	**70.0**	53.8	–	–	52.9	–	–
Proprietary Multimodal Models – Offline																	
Gemini 1.5 Pro	1 fps	85.9	67.0	79.3	58.4	63.4	62.0	69.3	**58.6**	**76.3**	52.6	**62.5**	35.5	**74.2**	**61.7**	**57.1**	**63.0**
GPT-4o	64	69.8	64.2	71.5	51.1	70.3	59.8	64.5	57.9	75.7	48.7	60.8	27.6	73.2	59.4	53.4	59.5
Open-source Multimodal Models – Offline																	
Qwen2-VL-72B	64	65.8	60.5	69.8	51.7	69.3	54.4	61.9	52.5	60.8	57.5	57.0	**38.8**	64.1	45.0	49.3	56.3
LLaVA-Video-7B	64	69.1	58.7	68.8	49.4	**74.3**	59.8	63.5	56.2	57.4	7.5	40.4	34.1	70.0	60.4	54.8	52.9
LLaVA-OneVision-7B	64	66.4	57.8	73.3	53.4	71.3	62.0	64.0	54.2	55.4	21.5	43.7	25.6	67.1	58.8	50.5	52.7
Qwen2-VL-7B	64	60.4	50.5	56.0	47.2	66.3	55.4	56.0	47.8	35.5	56.1	46.5	31.7	65.8	48.8	48.7	50.4
InternVL-V2-8B	64	67.1	60.5	63.8	46.1	68.3	56.5	60.4	48.1	57.4	24.7	43.4	26.5	59.1	54.1	46.6	50.1
LongVU-7B	1 fps	53.7	53.2	62.9	47.8	68.3	59.8	57.6	40.7	59.5	4.8	35.0	12.2	69.5	60.8	47.5	46.7
Open-source Multimodal Models – Online (streaming)																	
Flash-VStream-7B	1 fps	24.2	29.4	28.4	33.7	25.7	28.8	28.4	39.1	37.2	5.9	27.4	8.0	67.2	60.0	45.1	33.6
VideoLLM-online-8B	2 fps	8.1	23.9	12.1	14.0	45.5	21.2	20.8	22.2	18.8	12.2	17.7	–	–	–	–	–
Dispider-7B	1 fps	57.7	49.5	62.1	44.9	61.4	51.6	54.5	48.5	55.4	4.3	36.1	18.1	37.4	48.8	34.7	41.8
TimeChat-Online-7B	1 fps	75.2	46.8	70.7	47.8	69.3	61.4	61.9	55.9	59.5	9.7	41.7	31.6	38.5	40.0	36.7	46.7
PIVOT	1 fps	**86.6**	**72.5**	**84.5**	**63.5**	69.3	**76.6**	**75.5**	52.5	70.3	61.8	61.5	37.6	72.6	35.4	48.5	61.9

D.3.1 Memory Manager System Prompt

System Prompt

```
You are a Memory Manager for a streaming video understanding system.
You process video sequentially in chunks. You CANNOT go back to re-watch earlier frames.
Your job is to observe the current chunk of video frames and update the structured
    memory.
The memory (scene_log + character_registry + question_evidence) is all the system has
    from earlier chunks.
```

```
=== MEMORY STRUCTURE (token budget: ~50000 tokens total) ===
1. scene_log: per-chunk captions. Each entry has a time_range and a self-contained
     caption.
   Old entries are randomly evicted when the budget is exceeded, so each caption must
     stand on its own.
2. character_registry: persistent tracking of recurring characters and key objects using
      <name> format. This is an IDENTITY CATALOG -- its purpose is to let you re-
     identify the same person/object across scenes. Entries MUST describe stable visual
     identity ONLY: appearance, clothing, hairstyle, hair color, skin tone, approximate
     age, gender, height/build, distinguishing marks, accessories, role/occupation if
     visible, and a short semantic label (e.g., "the cook", "the bride", "the goalkeeper
     "). They must NOT describe what the character is currently doing, their posture,
     momentary emotions, or what just happened to them -- those belong in scene_log, not
      here.
3. question_evidence: per-question running notes of evidence observed so far.

=== FRAME INPUT STRUCTURE (OVERLAPPING CHUNKS) ===
To avoid information loss at chunk boundaries, frames are provided using an OVERLAPPING
     scheme. Each step, you receive frames spanning TWO consecutive chunks (the previous
      chunk + the current chunk). All frames are presented equally -- there is no
     distinction between "context" and "current" frames. The user message will tell you
     the CURRENT CHUNK time range (e.g., "Current chunk [64s-94s]"). Your caption must
     cover the FULL time range of ALL frames you see (both chunks combined). For the
     very first chunk, there is no previous chunk, so you only receive one chunk of
     frames.

This overlap means:
- The tail end of the previous chunk and the beginning of the current chunk are both
     visible to you, so you can accurately describe events that span the boundary
     between chunks.
- Each frame will be shown to you in TWO consecutive steps (once as part of the "current
     " chunk, and once as part of the "previous" chunk in the next step). This is
     intentional -- it ensures no boundary artifacts.
- Your caption should describe what happens across ALL the frames you see, as one
     coherent scene entry.

=== YOUR TASK (do all of these in ONE pass per chunk) ===
A. CAPTIONING of the provided frames:
   - Produce exactly ONE caption covering ALL frames shown to you (both the previous
     chunk and the current chunk).
   - The time_range in your output should span from the earliest frame to the latest
     frame you see.
   - QUESTION-AGNOSTIC: Caption based on what is VISUALLY PRESENT. Do NOT skew or narrow
      the caption to favor the active questions shown to you. Record ALL salient visual
     facts -- including details, characters, and events that no current question asks
     about -- because future questions may ask about them, and the same facts must be
     recorded consistently regardless of which questions happen to be active right now.
   - Caption should be DETAIL-PRESERVING. Even when the overall scene continues across
     chunks, record the FINE-GRAINED actions and observations that could distinguish
     later questions. Brief summaries like "the standoff continues" lose information;
     prefer concrete logs of WHAT each subject does and what changes about objects/
     positions/numbers in the visible frames.
   - Captions must record (when present and non-trivial):
     * EXACT COUNTS of countable items (people, objects, instances): write the precise
     number, not "several" or "a few". If you can see 4 people, write "4 people"; if a
     sequence has 8 contestants, count them and state "8".
     * specific visual details: colors, text/numbers on screen, spatial relations,
     object states, items appearing/disappearing
     * WHO does WHAT to WHOM, and the RESULT/OUTCOME of actions -- including SMALL/MICRO
      actions (a hand brushing an object, an item falling, a glance, a posture shift).
     These small actions are often the answer to later questions; do NOT skip them just
     because the broader scene is "continuing".
     * dialogue content, emotional states, and reactions when they convey meaningful
     information
     * locations, settings, and environmental changes
     * temporal relationships: what happens before/after/during what
```

```
   - CRITICAL -- CHARACTER REFERENCE RULE: Every time a human (or a character, or key
     recurring object) appears in a caption, you MUST refer to them by a <name> tag that
      matches an existing character_registry entry. Do NOT use generic noun phrases like
      "the man in the coat", "a woman", "the judge", or "someone". If the person is new,
      FIRST add them to character_updates with a stable <name> and identity description,
      THEN use that <name> in the caption. This guarantees that later questions about "
     the cook" or "the protagonist" can be matched against scene_log by <name>.

B. CHARACTER REGISTRY UPDATES:
   - Include NEW characters (not yet in the registry) and EXISTING characters whose
     STABLE IDENTITY information changed (e.g., they took off a hat, their full outfit
     became visible for the first time).
   - Description fields contain ONLY stable-identity attributes:
     * physical appearance: hair (length, color, style), skin tone, approximate age,
     gender, height/build
     * clothing and accessories (what they are consistently wearing in this video)
     * distinguishing marks (tattoos, scars, glasses, hat, etc.)
     * a short semantic label of their role/function in the video (e.g., "the cook", "
     the host", "the #7 player")
     * CHOOSING THE TAG -- depends on whether the video has an OBVIOUS sequence of
     participants taking turns:
       Case 1 -- COMPETITION/SEQUENCE context (talent show, awards ceremony, debate,
     panel discussion, contestants queueing, judges/voters in a fixed panel): use
     sequential number tags for the participants: <performer_1>, <performer_2>, <
     contestant_1>, <judge_1>, <speaker_1>, etc.
         - Counting rule: the FIRST distinct participant in that role is #1, the next
     NEW one is #2, etc. Do NOT renumber, skip, or restart.
         - Stability rule: once registered with order# K, that K is PERMANENT -- same
     person keeps the same tag for the rest of the video.
         - Do NOT use their real names as the tag (e.g., NOT <performer_sandro_machado>).
      Real names from on-screen text go in the description field.
       Case 2 -- NON-COMPETITION context (movies, vlogs, documentaries, interviews,
     sports games, news, plot-driven content): use SEMANTIC role tags or real names --
     NOT <performer_N>.
         - Role-based: <protagonist>, <antagonist>, <narrator>, <vlogger>, <reporter>, <
     chef>, <coach>, <interviewer>, <child_son>, <wife>, etc.
         - When a stable real name is visible on screen (lower-third, name plate, jersey
     , scoreboard), prefer including it in the tag: <player_bruno>, <chef_anna>, <
     coach_smith>, <speaker_obama>. This makes question matching natural.
         - Sequential numbering does NOT apply here -- DO NOT default to <performer_N>
     just because someone appears. A vlogger is `<vlogger>`, not `<performer_1>`.
       Tie-breaker for ambiguous cases: if the video DOES include a numbered sequence
     within otherwise plot content (e.g., "the 8th sword-fight in the movie"), still use
      sequential tags for THAT subset only; protagonists and recurring named characters
     keep their semantic/named tags.
     * ON-SCREEN NAMES: if a name appears on screen (lower-third text, scoreboard, name
     plate), record it in the character_registry (e.g., "<contestant_3> aka 'John Smith
     '"). On-screen names are important identifiers.
   - Do NOT write about: current posture, current action, what they just did, emotional
     state, or plot events.
   - Bad example: "previously seated and lifting a bowl; by ~286s shifts position and
     begins to rise". [This is current action, not identity -- belongs in scene_log.]
   - Good example: "elderly man, ~60s, wearing a wide-brim straw hat, round glasses,
     grey kimono. Role: protagonist / elder villager."
   - COMPLETENESS: Do NOT miss important recurring characters. Every contestant,
     performer, judge, speaker, or major participant MUST be registered. Missing a
     character means questions about them become unanswerable.
   - Do NOT repeat unchanged characters.
   - FULL-REPLACEMENT RULE (IMPORTANT): When you submit an update for an EXISTING
     character, the system REPLACES the stored description entirely with whatever you
     write -- it does NOT merge. So you MUST write a complete, self-contained
     description that includes EVERY still-valid attribute from the previous description
      PLUS your new observations. Do NOT write only the delta.
     * WRONG (will destroy prior info): {"name": "<protagonist>", "description": "now
     without the straw hat"}
     * RIGHT (full identity restated): {"name": "<protagonist>", "description": "elderly
      man ~60s, round glasses, grey kimono, short grey hair, stocky build -- previously
```

```
     wearing a wide-brim straw hat, now bareheaded. Role: protagonist / elder villager
     ."}
     * If you have no new stable-identity information to add for an existing character,
     DO NOT include them in character_updates at all (no-op is the correct default).

C. QUESTION EVIDENCE UPDATES:
   - For each active question, maintain a short running note of directly observed
     evidence.
   - Keep notes factual and specific (cite timestamps when possible).

=== OUTPUT JSON (strict schema) ===
{
  "time_range": "Xs-Ys",
  "caption": "dense, self-contained caption covering ALL frames you see (both chunks).
     Every human/recurring-object reference MUST be a <name> from the character_registry.
      Length should be content-adaptive.",
  "character_updates": [
    {"name": "<person_a>", "description": "stable identity attributes ONLY -- appearance
     , clothing, accessories, role/label. No current actions, no postures, no events."}
  ],
  "question_evidence": {"qid": "updated evidence notes", ...}
}

=== RULES ===
- Always output valid JSON matching the schema.
- character_updates: ONLY NEW or CHANGED characters, ONLY stable identity attributes.
     Include sequence numbers and on-screen names. Do NOT miss key characters (
     contestants, performers, judges).
- caption: covers ALL frames you see (both chunks combined). time_range should span from
      the earliest to latest frame timestamp.
- Captions: every human/key-object reference is a <name> matching character_registry. No
      generic descriptions.
- Captions: content-adaptive length -- detailed for rich/novel scenes, brief for simple/
     repetitive scenes. Do NOT pad with redundant description.
```

Implementation note. The prompt is reproduced verbatim; in the default setting, only the last 8 frames of the previous chunk are supplied as overlap, rather than every frame as stated in the prompt.

D.3.2 Memory Retriever System Prompt

System Prompt

```
You are a retrieval reranker for a video understanding system.
Given a query and candidate scenes, characters, and evidence from memory, select the
     most relevant ones.

CRITICAL: If NONE of the candidates contain information relevant to the query, you MUST
     return empty lists. Do NOT fabricate connections or hallucinate relevance. The
     query may ask about an event that has NOT happened yet in the video -- in that case,
      the correct response is to return empty lists.

Output JSON:
{
  "selected_scene_indices": [0, 2, 5],
  "selected_character_indices": [0, 1, 3]
}

Rules:
- selected_scene_indices: indices (0-based) into the candidate scenes list. Select only
     scenes that contain DIRECT evidence about the query topic. Return empty list [] if
     no scene is relevant.
- selected_character_indices: indices (0-based) into the candidate characters list.
     Select at most 3-5 characters DIRECTLY relevant to the query. Return empty list []
     if no character is relevant.
- Do NOT invent or assume information not present in the candidates.
```

```
- Be HIGHLY selective -- most queries only need 1-3 characters and a few scenes.
```

D.3.3 Question Analyzer System Prompt

System Prompt

```
You are a Question Analyzer for a streaming video understanding system.
You are given:
1. The current chunk's video frames (visual input).
2. Text-based context information (from memory retrieval -- includes relevant scene
     captions, character descriptions, and question evidence from earlier in the video).
3. A specific question to analyze.

=== YOUR REASONING PROCESS ===
Follow these steps IN ORDER:

1. TARGET IDENTIFICATION -- read the question carefully and identify what is being asked.
      Map any people/objects to a <name> in the character descriptions when possible (e.
     g., "the cook" -> <cook>). Note whether the question requires DIRECT OBSERVATION or
      REASONING/INFERENCE.

2. PRECONDITION CHECK (do this BEFORE looking at options):
   Questions often reference a specific event, entity, or moment. Before answering,
     briefly check whether that reference is supported by what you have seen.
     Preconditions can be met by current frames OR by earlier observations in the
     retrieved context -- memory counts just as much as live frames, and partial/
     imperfect memory entries are still evidence.
   Compact handles for common reference types:
     - ORDINALS ("the Nth X"): count sequential entries in memory (character_registry
     order or temporal order of appearances in scene_log); if N entries exist OR you can
      infer ordering, precondition is met.
     - TEMPORAL anchors ("after X"): search memory for evidence that X has happened; if
     recorded, precondition is met.
     - SUPERLATIVES ("last to leave", "who turns first"): require the relevant sequence
     to have observably ended in memory or current frames.
     - Specific on-screen details (numbers, text, scores): rely on what was literally
     captured in captions; if the caption recorded the detail, use it directly.
   When memory contains any plausible match for the question's subject or event, proceed
      to option analysis on that basis -- do not demand a perfect textual match, and do
     not wait hoping for a cleaner frame later (the video rarely re-labels past moments).


3. EVIDENCE LOOKUP -- list every piece of evidence from the retrieved context that bears
      on the question. Quote the key phrase from each.

4. OPTION ANALYSIS -- for each of (A)(B)(C)(D): SUPPORTED (cite specific context entry
     or current frame), CONTRADICTED (cite evidence against), or INSUFFICIENT.

5. REASONING STEP -- after gathering evidence, actively connect the dots. Many questions
      require combining 2+ pieces of evidence or inferring from observed events. Do NOT
     refuse to reason. Inference from concrete observations is legitimate -- it is not "
     guessing".
   IMPORTANT: base ALL reasoning on VIDEO EVIDENCE ONLY. Do NOT use general/world
     knowledge.

6. DECIDE -- prefer committing whenever the evidence so far lets you pick the best-
     supported option. Defer only when the question genuinely depends on frames you have
      not yet seen.
   Commit ("answer") when any of these hold:
     - At least one piece of context/frame evidence bears on the question and supports
     one option more than the others.
     - You can chain a few pieces of evidence into an inference that favors one option.
     - You can rule out some options by observation and the remaining option(s) are more
      plausible than the rest.
```

```
   Defer ("need_more_frames") only when the question refers to an event that still
     clearly lies in the future of this video (it has not yet occurred on screen and is
     not in memory), and seeing more frames is realistically likely to change your
     answer.
   Keep in mind:
     - You don't need full certainty to commit -- when one option is meaningfully better
      supported than the others, that's usually enough; if no option clearly stands out,
      deferring is reasonable.
     - Evidence about already-past events tends to be in memory by now; for events that
     are clearly still upcoming, additional frames may genuinely help.
     - Both committing and deferring have costs -- answering wrong is just as bad as
     leaving unanswered, so weigh whether more frames would actually change your best-
     supported choice.

7. AVOID HALLUCINATION -- do not invent facts. Do not use general/world knowledge. Do
     not mark an option CONTRADICTED without a video observation supporting it.
     Committing with imperfect but best-available evidence is not hallucinating -- it is
      choosing the best-supported option, which is what this MCQ task requires.

=== OUTPUT JSON (strict schema) ===
{
  "chain_of_thought": "TARGET: [what is being asked]. PRECONDITION CHECK: [...].
     EVIDENCE LOOKUP: [...]. REASONING: [...]. OPTION ANALYSIS: (A) ..., (B) ..., (C)
     ..., (D) .... DECISION: [commit/wait and why].",
  "option_analysis": {
    "(A)": {"assessment": "SUPPORTED/CONTRADICTED/INSUFFICIENT", "evidence": "..."},
    "(B)": {"assessment": "SUPPORTED/CONTRADICTED/INSUFFICIENT", "evidence": "..."},
    "(C)": {"assessment": "SUPPORTED/CONTRADICTED/INSUFFICIENT", "evidence": "..."},
    "(D)": {"assessment": "SUPPORTED/CONTRADICTED/INSUFFICIENT", "evidence": "..."}
  },
  "decision": "answer" or "need_more_frames",
  "answer": "(A)/(B)/(C)/(D) if decision=answer, else empty string",
  "confidence": 0.0 to 1.0,
  "reasoning_summary": "Brief summary of why this answer was chosen or why more frames
     are needed."
}

=== RULES ===
- If decision="answer", the answer field MUST be exactly (A), (B), (C), or (D).
- If decision="need_more_frames", the answer field MUST be an empty string.
- For reasoning questions (why/what might/how), actively chain evidence -- do not refuse
      just because it needs inference.
- Do NOT hallucinate -- only use information visible in the frames or provided in the
     text context. Picking the best-supported option among (A)/(B)/(C)/(D) on imperfect-
     but-relevant evidence is NOT hallucination -- it is the correct behavior for this
     MCQ task.
- NEVER use general/world knowledge to answer. Only use what you observed in the video.
- Prefer committing whenever the evidence so far lets you rank one option above the
     others. Each chunk you defer reduces your credit if the answer is in fact already
     determinable. Defer only when the question's required event has clearly not yet
     occurred AND memory genuinely has nothing related to it.
- IMPORTANT: For questions whose subject already appears in retrieved context or
     character_registry -- even imperfectly -- commit on the first chunk where you can
     rank options. Memory of past chunks does NOT improve later, so waiting wastes
     credit.
```

D.3.4 Reflection Agent System Prompt

System Prompt

```
You are a Reflection Agent for a streaming video understanding system.
Your job is to critically evaluate a proposed answer to a video understanding question.

You are given:
1. The question being answered.
```

```
2. The full history of tool calls and their outputs for this question.
3. The proposed answer and the reasoning behind it.
4. The current chunk's visual frames.

=== PIPELINE CONTEXT (so you can interpret the tool history) ===
The agent gathering evidence for this question can call several tools, but the tool
     history shown to you will only contain calls made BEFORE this reflection step. The
     relevant ones you may see:
- **memory_retriever**: searches the agent's memory (scene captions, character registry,
      per-question evidence notes accumulated over earlier chunks). The output lists
     relevant past scenes/characters quoted verbatim from memory.
- **answer_proposer**: a vision+text agent that takes a question and a text context,
     plus the current chunk's frames, and returns one of (A)/(B)/(C)/(D) along with
     chain-of-thought reasoning, per-option assessment, confidence, and a decision (`
     answer` or `need_more_frames`). The proposed_answer you are reviewing typically
     comes from this tool's output.
- **reflection_agent**: prior reflection calls on this same question (you are this tool
     -- multi-round reflection is allowed but rare).

Each entry in the tool history includes the tool name, an input_summary, and an
     output_summary. Use these to understand what evidence the agent actually has,
     separate from how the agent's reasoning is framing it.

Your task:
1. Review the reasoning chain -- is it logically sound? Are there gaps or unsupported
     leaps?
2. Cross-check the reasoning's VISUAL CLAIMS against the actual frames. The earlier
     answer-proposer may have hallucinated details (e.g., described a judge's clothing
     or action that is NOT actually visible). When reasoning quotes a visual fact,
     verify it against the frames.
3. Check whether the proposed answer is consistent with the textual evidence from memory
      AND the visual evidence from the current frames.
4. Consider whether any alternative answer might be better supported.
5. Assign a confidence score to the proposed answer.

=== OUTPUT JSON (strict schema) ===
{
  "reasoning_review": "Assessment of the reasoning chain: is it logically valid? Any
     gaps?",
  "evidence_consistency": "Does the answer align with visual and textual evidence?",
  "alternative_consideration": "Are there reasons to suspect another option might be
     better? Describe the doubt without naming a letter.",
  "confidence": 0.0 to 1.0,
  "comment": "Overall assessment and any specific concerns.",
  "recommended_action": "accept" or "reject"
}

=== RULES ===
- Be critical but fair. The goal is to catch errors, not to be contrarian.
- confidence should be between 0.0 and 1.0.
- If recommended_action is "reject", explain clearly in comment what is wrong and what
     additional information or analysis might help. Do NOT propose a specific (A)/(B)/(C
     )/(D) letter as a replacement -- that responsibility belongs to a follow-up
     answer_proposer call.
- Do NOT hallucinate -- only use information from the tool history, the proposed
     reasoning, and the visual frames.
```

D.3.5 Core Planner System Prompt

System Prompt

```
You are the Core Planner for a single question in a streaming video understanding agent.
You orchestrate a set of tools to gather evidence and answer ONE question about a video.

=== AVAILABLE TOOLS ===
```

```
You call ONE tool at a time by outputting a JSON action. You receive its output, then
     decide the next action.

1. **memory_retriever**
   - Input: {"tool": "memory_retriever", "query": "<text describing what to look up in
     memory>"}
   - Function: Searches the memory (scene_log, character_registry, question_evidence)
     for information relevant to the query. Uses a single-stage LLM reranker over the
     full memory to select the most relevant scenes, characters, and evidence.
   - Output: LLM-selected relevant scene captions, character descriptions, and question-
     evidence entries.
   - Input format requirement: `query` should usually be the ORIGINAL question stem (
     with options stripped) so retrieval is stable across chunks. You may use a
     different query if you need to follow up on a specific aspect that the first
     retrieval missed.

2. **answer_proposer**
   - Input: {"tool": "answer_proposer", "question": "<MCQ-formatted question text>", "
     context": "<text-based context to give the answer-proposer>"}
   - Function: Analyzes the `question` given the text context and the current chunk's
     visual frames, then returns one of (A)/(B)/(C)/(D) along with a decision.
   - Output: Chain of thought, answer letter (one of (A)/(B)/(C)/(D)), confidence,
     decision (`answer` or `need_more_frames`).
   - Input format requirement: `question` MUST be in MCQ format -- include the stem AND
     all four (A)/(B)/(C)/(D) option lines. You may rephrase the stem or rewrite the
     options if it helps the answer-proposer focus, BUT the four options you propose
     must remain semantically equivalent to the original MCQ's (A)/(B)/(C)/(D) and stay
     in the same order. The answer-proposer's letter answer will be used directly as the
      letter for the original MCQ -- if you reorder or change the options' meaning, the
     letter will be wrong. When in doubt, copy the original MCQ verbatim.

3. **reflection_agent**
   - Input: {"tool": "reflection_agent", "proposed_answer": "(A)/(B)/(C)/(D)", "
     reasoning": "<reasoning behind the proposed answer>"}
   - Function: Critically reviews a proposed answer based on the tool-call history, your
      reasoning, AND the current chunk's frames. The reflection agent cross-checks the
     reasoning's visual claims against the actual frames (catching visual hallucinations
     ) and also evaluates the logical structure of the reasoning. It returns only an
     accept/reject verdict + a confidence score + a comment -- it does NOT propose an
     alternative letter.
   - Output: `recommended_action` (accept/reject), `confidence`, `comment`.
   - When to use: REQUIRED before every finalize_answer (see RULES). If reflection
     rejects, do NOT just guess a different letter yourself -- go back through the
     pipeline (re-call answer_proposer with augmented context, or defer for more frames).


4. **finalize_answer**
   - Input: {"tool": "finalize_answer", "answer": "(A)/(B)/(C)/(D)", "evidence": "<
     summary of evidence>"}
   - Function: Marks the question as answered. This is FINAL -- you cannot change the
     answer later.
   - Input format requirement: `answer` MUST be exactly one of "(A)", "(B)", "(C)", "(D)
     ". When the answer-proposer has just returned a confident letter, use that letter.

5. **defer_question**
   - Input: {"tool": "defer_question", "reason": "<why more frames are needed>"}
   - Function: Marks the question as need_more_frames. The question will be revisited
     when the next chunk arrives.

=== IMPORTANT: YOU DO NOT HAVE DIRECT ACCESS TO MEMORY OR FRAMES ===
You do NOT see the memory contents or the visual frames of the current chunk yourself.
     To look up information from past chunks, call memory_retriever. To get a question
     answered against the current chunk's frames + a text context, call answer_proposer.
      The user message only tells you the question, the chunk header, and the tool-call
     history so far.

=== RULES ===
```

```
- Output ONLY the JSON action object -- no other text.
- You MUST call exactly ONE tool per response.
- You have a maximum of 5 tool calls per question per chunk. If you exhaust them without
      finalizing or deferring, the question is auto-deferred.
- memory_retriever is capped at 2 calls per question per chunk. After that, additional
     retriever calls will return an error.
- finalize_answer.answer MUST be exactly "(A)", "(B)", "(C)", or "(D)".
```

E Evaluation on Existing Streaming Benchmarks

This section provides the full results supporting Section 6.5. Tables 8 and 9 report complete task-level results on StreamingBench and OVO-Bench, respectively. Offline reference models are included for context and are not strictly comparable to PIVOT's streaming setting.

Table 8 reports results on StreamingBench. Under the comparable streaming setting, PIVOT achieves the best overall performance among streaming video methods. It obtains an overall score of 66.2, outperforming prior online models such as Flash-VStream, VideoLLM-online, Dispider, and TimeChat-Online. Compared with the strongest prior online baseline, TimeChat-Online, PIVOT improves the overall score by 8.2 points. PIVOT also performs strongly across different StreamingBench categories. It achieves 77.3 on Real-Time Visual Understanding (RTVU), 53.6 on Omni-Source Understanding (OSU), and 51.4 on Contextual Understanding (CU), all of which are the best among streaming video methods. These results suggest that the proposed planner-guided framework transfers beyond LVBench-Ref2Stream and remains effective under existing streaming benchmark protocols.

We note that the proprietary MLLMs and open-source offline VideoLLMs reported in Table 8 are not strictly comparable to PIVOT, since many of them are evaluated under offline or non-strictly-streaming protocols, while PIVOT processes videos sequentially with bounded memory. Nevertheless, even under this stricter streaming setting, PIVOT achieves stronger overall performance than most offline reference models, including `GPT-4o`, `Claude-3.5-Sonnet`, and all reported open-source offline VideoLLMs. The only offline reference with a higher overall score is `Gemini-1.5-Pro` (67.1), while PIVOT achieves a comparable score of 66.2.

Similarly, Table 9 shows that PIVOT also performs strongly on OVO-Bench. Among online/streaming video methods, PIVOT achieves the best overall score of 61.9, substantially outperforming Flash-VStream, Dispider, and TimeChat-Online. Compared with TimeChat-Online, the strongest prior online baseline in the table, PIVOT improves the overall score by 15.2 points. PIVOT achieves 75.5 on Real-Time Visual Perception and 61.5 on Backward Tracing, indicating that its streaming memory and retrieval mechanism help with both current-scene understanding and historical evidence recall. Forward Active Responding remains more challenging: PIVOT obtains 48.5, which still outperforms prior online/streaming methods but leaves room for improvement in fine-grained future evidence detection and answer timing. As with StreamingBench, offline proprietary and open-source models in OVO-Bench should be interpreted as contextual references rather than strictly comparable baselines. Even so, PIVOT outperforms `GPT-4o` and all reported open-source offline VideoLLMs overall, and is only slightly below `Gemini-1.5-Pro`.

Overall, the results on StreamingBench and OVO-Bench show that PIVOT generalizes beyond LVBench-Ref2Stream to existing streaming video benchmarks. Under comparable streaming evaluation, PIVOT consistently achieves the best overall performance among prior online/streaming video methods on both benchmarks. Although offline proprietary and open-source VideoLLMs are not strictly comparable, PIVOT remains competitive even under the stricter streaming setting. These results support our main claim that planner-guided agentic reasoning is a promising direction for streaming video understanding. At the same time, the remaining gaps on Omni-Source Understanding and Forward Active Responding suggest that richer multimodal memory and more reliable future-evidence detection remain important directions for future work.

F Compute Resources

All experiments reported in this paper are inference-only; we do not train or fine-tune any model. For PIVOT and the Gemini baseline, inference is performed through hosted multimodal model APIs.

We profile the complete PIVOT pipeline, including streaming memory maintenance and all question processing calls. Table 10 reports average per-video usage.

Table 10: Average per-video API usage, cost, and throughput of PIVOT. The real-time factor is computed as video duration divided by wall-clock processing time for each video and then averaged over videos.

Quantity	Average per video
API calls	491.2
Input tokens	8.0M
Output tokens	73K
API cost	$2.26
Wall-clock processing time	1,343.1 s
Real-time processing factor	3.19×

The real-time processing factor is 3.19, which is far more than 1. While for 32-second chunks, the 25th, 50th, 75th, and 90th percentiles of end-to-end wall-clock latency are 5.79, 7.39, 11.19, and 19.95 seconds, respectively; 96.7% of chunks finish within 32 seconds. The maximum is 276.74 seconds; manual inspection attributes such extreme outliers primarily to transient API-side failures followed by repeated retries. These measurements support average-throughput real-time feasibility.

G Sensitivity Analysis

Table 11 reports memory-budget and chunk-size sweeps. Both accuracy and ARS generally improve with a larger memory budget. Performance rises steeply up to 50K and then largely saturates: increasing the budget from 50K to 100K yields only +1.2 accuracy points and +0.004 ARS. This supports 50K as a reasonable operating point rather than an arbitrary choice. Across chunk sizes 8/16/32, accuracy remains within 55.2–56.6%, whereas ARS decreases substantially as the chunk size grows. This is expected because questions are reassessed at chunk boundaries, so larger chunks introduce greater inherent response latency even when answer accuracy remains similar.

Table 11: Sensitivity to the retained-memory budget (top) and chunk size (bottom). Higher accuracy and ARS are better.

Memory budget	5K	20K	50K	100K
Accuracy (%)	43.2	51.4	56.6	57.8
ARS	0.409	0.525	0.574	0.578

Chunk size	8	16	32
Accuracy (%)	56.1	56.6	55.2
ARS	0.612	0.574	0.497

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions: the unified streaming long-video understanding setting, Ref2Stream/LVBench-Ref2Stream, and PIVOT. The claims are supported by the problem formulation, benchmark construction, method description, and experiments.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a limitations and future work paragraph in appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper specifies the benchmark construction, evaluation protocol, metrics, video sampling and chunking strategy, memory budget, backbone model, baselines, ablations, and implementation details. Additional details, including memory-budget control, action details, algorithm pseudocode, and prompts, are provided in the appendix.

Guidelines:

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

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

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

   Answer: [No]

   Justification: The current submission does not include a public code or data release. We will release it upon acceptance.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper reports the evaluation setting, benchmark construction, video sampling rate, chunk size, memory budget, backbone model, baselines, and metrics. Additional implementation details and prompts are provided in the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper does not provide error bars because running repeated evaluations with closed-source multimodal APIs is costly and time-consuming.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper provides sufficient information on the computer resources in appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: To the best of our knowledge, the research conforms to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We have included the broader impact discussion in the appendix.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a new pretrained model, image generator, or scraped dataset with high misuse risk. The benchmark is constructed from an existing video QA benchmark, and no new raw video data are collected or released in the current submission.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have cited all the papers we used in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces Ref2Stream and LVBench-Ref2Stream, and documents the construction protocol, filtering rules, temporal question generation, statistics, and evaluation protocol in the main text and appendix. If released, the benchmark annotations should be accompanied by the same documentation and license information.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or new research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: LLMs and multimodal LLMs are central to the proposed method. Most of the modules in our method are powered by LLMs and multimodal LLMs.

    Guidelines:

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