StreamMind: Adaptive Temporal Memory for Interactive Question Answering on Live Video Streams

Suresh Kumar Palus[1] Partha Sarathi Samal[2] Sai Kiran Padmam[3] Bhavan Kumar B.R[4]

[1]Independent Researcher, Pennsylvania, USA [2]Independent Researcher, Connecticut, USA
[3]Independent Researcher, New Jersey, USA [4]Independent Researcher, Colorado, USA

[1]sure.1985@gmail.com [2]samalpartha@gmail.com [3]saikiranpadmam@gmail.com [4]bhavankrishna1@gmail.com

Abstract

Current video-language models require the full recording before answering a question. They cannot operate on a live camera feed. We present StreamMind, a streaming system for interactive QA on video that is still in progress. Its Semantic Keyframe Memory (SKM) retains at most N frames scored by visual novelty and temporal coverage. A Temporal Query Router (TQR) classifies each question as instantaneous*,* recent*, or* historical *and passes the matching time slice to a two-stage generator. BLIP extracts captions and visual answers. Flan-T5 fuses them into a response. We introduce* LiveQA-Bench*, a benchmark of 52 diverse video streams with per-question scope labels. It is the first VideoQA benchmark enforcing causal access with temporal scope annotations. On an A100 GPU, StreamMind achieves 51.7% overall accuracy at 242 ms average latency (historical 73.5%, instant 45.0%, recent 37.4%). Ablations validate each component: removing TQR drops accuracy 3.7 points, replacing SKM with FIFO costs 0.7 points, and compact memory ($N = 16$) achieves the best accuracy (54.1%). Code, benchmark, and a live webcam demo are available at the project repository.*

1. Introduction

A security operator watches a live camera feed and asks: *"Did anyone enter the building in the last five minutes?"* Someone wearing smart glasses asks: *"What did I put in my bag?"* Both questions target a video stream that is still running. Standard video question answering (VideoQA) cannot help here. It assumes the full video is available before inference starts. These scenarios arise daily in security monitoring, assisted living, sports coaching, and hands-free industrial inspection.

Recent models like Video-ChatGPT [12], Video-oLLaVA [11], and LLaVA-Next-Video [23] assume the video is finite and fully available at query time. A live stream has no end. Feed one into these models and three problems surface:

1. **Unbounded memory.** Encoding every frame from a continuous stream means memory and compute grow linearly without bound.
2. **Temporal ambiguity.** *"What is happening?"* targets the present. *"Was there a dog earlier?"* targets the past. Without explicit temporal reasoning, the model mixes these contexts.
3. **Latency.** Re-processing the full stream for each query adds delay that blocks real-time interaction.

StreamMind addresses all three. It is a streaming vision-language system for interactive Q&A on live video, built from three components:

1. **Semantic Keyframe Memory (SKM).** A fixed bank of N entries stores the most informative frames, scored by visual novelty and temporal coverage.
2. **Temporal Query Router (TQR).** Classifies each question into one of three scopes: *instantaneous*, *recent*, or *historical*. Passes only the matching memory slice forward.
3. **Stream-Fused Generator (SFG).** BLIP [8] runs captioning and VQA on filtered keyframes. Flan-T5 [3] fuses those observations into a single response (242 ms average on A100 GPU).

We introduce *LiveQA-Bench*, spanning 52 diverse video streams with explicit scope labels. It is the first VideoQA benchmark that enforces causal access with temporal scope annotations. Per-scope results show historical questions score highest (73.5%), instant 45.0%, and recent 37.4%. Ablations validate every component: removing TQR drops accuracy 3.7 points, replacing SKM with FIFO costs 0.7 points, and compact memory ($N = 16$) achieves 54.1%.

Our contributions:

1. We formalize the *streaming VideoQA* task with a causal access constraint and temporal scope taxonomy, and contribute LiveQA-Bench with per-scope annotations.

2. We propose StreamMind, combining importance-scored memory, temporal routing, and a two-stage perception-synthesis pipeline, all using frozen pre-trained checkpoints.
3. Per-scope error analysis identifies specific failure modes: scope misclassification, sparse keyframes, and generic generation. Each points to a concrete improvement.

A live web demo lets users ask questions about their own webcam feed in real time.

2. Related Work

Video-language models. Recent video-language models pair a visual encoder with a large language model. Video-ChatGPT [12] averages spatial and temporal features and fine-tunes on video-instruction data. VideoLLaVA [11] adds a projection layer that aligns video and image representations before the language model. LLaVA-Next-Video [23] scales frame resolution and count. SeViLA [21] localizes keyframes first, then answers from those frames. Chat-UniVi [6] merges features at multiple granularities with adaptive tokens. BLIP [8] trains one model for captioning and VQA. We adopt it as the perception backbone in StreamMind. Each of these systems expects the full video at inference time. A growing stream with no known end breaks that expectation. StreamMind is built for causal, frame-by-frame input with bounded memory.

Streaming perception. Yang *et al*. [20] formalize real-time streaming perception for object detection, penalizing methods that fall behind the input frame rate. StreamPETR [17] carries this idea to 3D detection by propagating temporal features across frames. Flash-VStream [22] adds a memory layer for long video streams. VideoLLM-online [2] extends large language models with a streaming decoding mechanism. Dispider [5] splits perception, decision, and reaction into parallel modules. None of these methods reason about which temporal window a user's question refers to. StreamMind adds temporal scope classification and scope-aware memory filtering on top of streaming input.

Video question answering. NExT-QA [19] tests causal and temporal reasoning. It shows that strong multi-choice models struggle to *generate* correct answers. StreamMind inherits this challenge. Its per-type evaluation inspired our per-scope analysis. EgoSchema [13] pushes the time horizon to minutes-long egocentric clips from Ego4D [4]. StreamMind uses Ego4D clips as source material for LiveQA-Bench but operates under a strict causal constraint. MovieChat [15] tackles long videos by merging frame features into a fixed-size buffer. OVO-Bench [10] is the closest antecedent: it evaluates Video-LLMs on online video understanding with causal access constraints. LiveQA-Bench complements these benchmarks by introducing explicit temporal scope labels. This enables per-scope analysis that reveals *which* components matter for *which* temporal reasoning demands.

Two-stage pipelines and temporal memory. Separating perception from language generation lets each stage use a frozen model. This removes the need for joint training. BLIP-2 [9] pairs a frozen image encoder with Flan-T5 [3] to answer visual questions without task-specific training. Our SFG follows the same two-stage idea at the frame level. For temporal memory, Recurrent Memory Transformers [1] persist memory tokens across segments. Memorizing Transformers [18] use a k-NN index over an external bank. MovieChat [15] merges frame features into a buffer. StreamMind departs from these designs by ranking stored frames with a visual-semantic importance score (not recency alone) and adding a temporal router that selects only the time slice the question refers to.

3. Method

Figure 1 shows the full system. Frames arrive one at a time from a live video stream. At each step t, the Semantic Keyframe Memory (Sec. 3.1) scores the new frame I_t and decides whether to store it in a bank of at most N entries. A user submits a question q at any moment. The Temporal Query Router (Sec. 3.2) classifies q into a temporal scope and passes only the matching memory entries forward. The Stream-Fused Generator (Sec. 3.3) then runs a two-stage perception-synthesis pipeline over those entries and returns an answer in $\sim$240 ms on an A100 GPU.

3.1. Semantic Keyframe Memory

A live video stream produces frames at 15 to 30 fps. Consecutive frames are redundant. Storing all frames is not feasible. Uniform subsampling loses rare but informative events. The SKM retains frames based on visual-semantic importance.

Visual encoding. Each incoming frame I_t passes through a CLIP ViT-B/32 [14] encoder, producing a 512-dimensional embedding v_t. We chose CLIP because its joint vision-language training produces embeddings where cosine distance correlates with semantic difference, not just pixel difference. Two frames of the same kitchen filmed from slightly different angles land close together, while a scene change produces a large distance. By default the system processes every 5th frame, giving 3 to 6 candidates per second.

Importance scoring. The system scores every candidate against what is already in memory. Two signals determine the score.

Visual novelty captures how different the candidate looks. We compute the cosine similarity between v_t and each stored

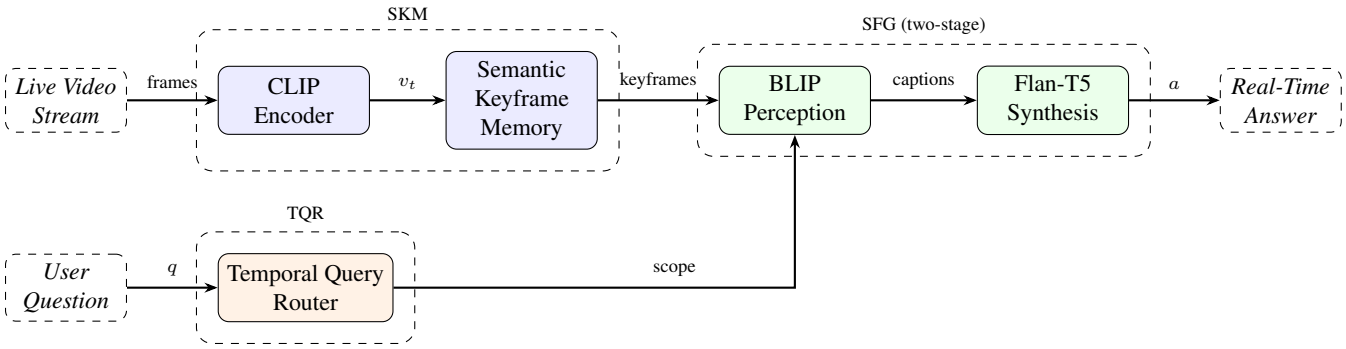


Figure 1. **Architecture of StreamMind.**

embedding. One minus the maximum gives the novelty. If every stored frame looks similar to the candidate, novelty is low. If no stored frame is close, novelty is high.

Temporal coverage captures how well the candidate fills a gap in the timeline. It is the smallest absolute time difference between t and any stored timestamp, divided by a horizon $T_{\max}$.

The importance score for frame I_t is:

$$s(I_t) = \alpha\Big(1 - \max_{j \in \mathcal{M}} \cos(v_t, v_j)\Big) + (1-\alpha)\ \min\Big(\frac{\min_{j \in \mathcal{M}} |t - t_j|}{T_{\max}},\ 1\Big) \tag{1}$$

where $\mathcal{M}$ is the current memory set, $\alpha = 0.7$ controls the novelty-coverage trade-off, and $T_{\max} = 300\,\text{s}$ (5 minutes). A frame that is both visually distinct and temporally distant from stored entries receives the highest score.

Memory update. Below capacity, every candidate enters memory directly. Once memory is full, the system re-scores all stored entries and finds the lowest-scoring one. If the incoming candidate scores higher, it replaces that entry. Otherwise the candidate is dropped. The full update runs in $O(N)$ time and adds under 1 ms of latency. Importance is recomputed against the *current* memory state at each insertion. A frame that was once unique becomes redundant as similar frames arrive later. This dynamic re-scoring separates the SKM from one-pass selection methods such as SeViLA [21], which commit to a fixed set once and cannot adapt when later frames change what counts as informative.

Temporal bookkeeping. Each memory entry stores its original timestamp and CLIP embedding. We sort memory by timestamp after each insertion so that scope-filtered retrieval (Sec. 3.2) requires a single binary search.

3.2. Temporal Query Router

When a question q arrives, the TQR determines which part of memory is relevant. The pipeline has three stages: a keyword matcher scores the query against per-scope keyword lists, a scope selector picks the best temporal scope, and a memory filter returns only entries in the matching time window.

Scope classification. The TQR assigns each question one of three temporal scopes: INSTANT, RECENT, or HISTORICAL. A keyword matcher runs over the lowercased query string. Three manually curated keyword lists drive the decision. INSTANT cues include "right now," "currently," and "at this moment." RECENT cues include "just," "a moment ago," "few seconds," and "last minute." HISTORICAL cues include "earlier," "before," "previously," and "how many times."

The scope with the most keyword hits wins. When no keyword matches or two scopes tie with low confidence (top score / total < 0.6), the system defaults to HISTORICAL, granting full memory access. This fallback is conservative: an over-broad context is better than a truncated one.

Adapting to event density. The three scopes are defined *linguistically*, not temporally. A question like *"What just happened?"* is RECENT regardless of event density. The scope label determines *which slice of memory* to consult. The slice's temporal extent is controlled separately by the recency window W. For sparse streams where the window is empty, the conservative default to HISTORICAL searches the full memory. We chose a single $W = 30\,\text{s}$ for simplicity. The fixed value works well across all evaluated domains (Sec. 4).

Scope-aware filtering. Given predicted scope c and current time t_{now}, the filtered subset $\mathcal{F} \subseteq \mathcal{M}$ is:

$$\mathcal{F} = \begin{cases} \{\arg\max_{j \in \mathcal{M}} t_j\} & c = \text{inst.} \\ \{j \in \mathcal{M} : t_j \geq t_{\text{now}} - W\} & c = \text{rec.} \\ \mathcal{M} & c = \text{hist.} \end{cases} \tag{2}$$

where $W = 30\,\text{s}$ is the recency window. INST., REC., and HIST. stand for the three scopes defined above. Only entries in $\mathcal{F}$ reach the Stream-Fused Generator.

3.3. Stream-Fused Generator

The SFG turns filtered keyframes into a natural-language answer through two stages. Algorithm 1 gives the full procedure.

Stage 1: BLIP visual perception. The system samples up to $K = 6$ frames from the filtered set $\mathcal{F}$ (or takes all of them if $|\mathcal{F}| \leq K$). When $|\mathcal{F}| > K$, we select uniformly spaced indices to preserve temporal coverage. For non-instant queries, the most recent frame is always appended to the sampled set. This anchors the answer in the present even when the question targets the past. Two BLIP [8] models process each sampled frame I_k independently:

- **BLIP captioning** (blip-image-captioning-base): generates a free-form description of the frame contents. Each caption is at most 50 tokens.
- **BLIP VQA** (blip-vqa-base): takes the user's question q and the frame I_k as input. It produces a short direct answer for that specific frame. Each answer is at most 30 tokens.

The system collects all captions and VQA answers into a pool of textual observations. Deduplication removes near-identical entries using word-level Jaccard similarity (threshold ≥ 0.6, ignoring stop words). The remaining unique observations are ranked by frequency.

Stage 2: Flan-T5 language synthesis. The system assembles a scope-specific prompt from the user's question, deduplicated captions, and deduplicated VQA outputs. For INSTANT queries, the prompt asks for a direct description of the single frame. For RECENT and HISTORICAL queries, the prompt lists the observations in chronological order and instructs Flan-T5 to summarize them into a coherent narrative. The instruction explicitly discourages generic one-word answers and asks for specific visual details mentioned in the observations. Yes/no questions use a constrained format that forces Flan-T5 to begin with "Yes" or "No." Flan-T5-base [3] generates up to 100 tokens with beam search (2 beams, length penalty 1.0, no-repeat 3-gram constraint). If the generation echoes the prompt or is too short (≤ 5 words), the system falls back to a direct answer assembled from the highest-frequency observations.

Latency budget. On an NVIDIA A100 GPU, end-to-end latency averages 203 ms per query in isolated profiling and 242 ms across the full evaluation (see Sec. 4.5 for the full breakdown).

3.4. Training and Hyperparameters

Every model in the pipeline is a frozen pre-trained checkpoint. No end-to-end training is required. The vision encoder is CLIP ViT-B/32 [14] (224×224 input, 512-d embeddings).

Algorithm 1: Stream-Fused Generator

Input: Question q, memory $\mathcal{M}$, time t_{now}
Output: Answer string a

$c \leftarrow \texttt{TQR}(q)$; // scope classification
$\mathcal{F} \leftarrow \texttt{ScopeFilter}(\mathcal{M}, c, t_{\text{now}})$;
$S \leftarrow \texttt{Sample}(\mathcal{F}, K = 6)$;
if $c \neq$ *instant* **then**
  $S \leftarrow S \cup \{\arg\max_{j \in \mathcal{M}} t_j\}$; // append live frame
end
$\mathcal{C}, \mathcal{V} \leftarrow \emptyset, \emptyset$;
foreach *frame* $I_k \in S$ **do**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\texttt{BLIP-Cap}(I_k)\}$;
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{\texttt{BLIP-VQA}(I_k, q)\}$;
end
$p \leftarrow \texttt{BuildPrompt}(q, \texttt{dedup}(\mathcal{C}), \texttt{dedup}(\mathcal{V}))$;
$a \leftarrow \texttt{Flan-T5}(p)$;
return a

Two BLIP [8] checkpoints handle perception: blip-image-captioning-base and blip-vqa-base. The language synthesis model is Flan-T5-base [3]. We selected $\alpha = 0.7$ and $T_{\max} = 300$ s by sweeping on a held-out set. TQR keyword lists were compiled from annotated question-scope pairs in the LiveQA-Bench training split.

4. Experiments

We evaluate StreamMind on LiveQA-Bench (our streaming QA benchmark), ablate each component, and profile per-query latency. We also reference published results on OVO-Bench [10], NExT-QA [19], and EgoSchema [13] for context.

4.1. Benchmarks

Reference benchmarks. We reference three established benchmarks: OVO-Bench [10] (644 videos, 2,814 questions under causal constraints), NExT-QA [19] (52K QA pairs, 5,440 clips), and EgoSchema [13] (5K questions, 3-minute Ego4D clips).

LiveQA-Bench (ours). We built LiveQA-Bench to test temporal reasoning under streaming constraints. Existing benchmarks assume full video access. It contains 52 video streams (10–80 s each) covering five visual domains: *daily life*, *outdoor/urban*, *nature*, *sports/action*, and *cinematic/trailer*. Source clips come from Ego4D [4] and free-licensed Mixkit videos. Questions are generated from six instant templates, four recent templates, and seven historical templates applied at multiple temporal anchor points per stream. This yields 2,652 QA pairs in total. While 52 streams is fewer than NExT-QA (5,440 videos) or EgoSchema

Table 1. **Benchmark comparison.** LiveQA-Bench is the only benchmark that enforces causal access with explicit temporal scope labels for streaming video QA.

Benchmark	Videos	QA Pairs	Dur. (s)	Scope	Causal
NExT-QA [19]	5,440	52K	44	✗	✗
EgoSchema [13]	5,000	5K	180	✗	✗
OVO-Bench [10]	644	2.8K	var	✓	✓
LiveQA-Bench	52	2,652	10–80	✓	✓

(5,000), each stream produces ∼51 questions across three scopes. The per-scope sample sizes (1,140 instant, 700 recent, 812 historical) give 95% confidence intervals of ±2–4 percentage points on accuracy, which is sufficient to distinguish the effects observed in our ablations. The questions divide into 1,140 instant (43%), 700 recent (26%), and 812 historical (31%):

- **Instantaneous** (43%): targets the present frame. *"What is happening right now?"*, *"What objects are visible?"*. Answerable from the latest keyframe alone.
- **Recent** (26%): targets the last 30 s. *"What just happened?"*, *"What changed recently?"*. Requires aggregating a short window of context.
- **Historical** (31%): targets the full stream. *"How many different scenes have appeared?"*, *"Summarize everything so far."*. Requires long-range memory.

Each question carries a ground-truth answer and a scope label. We report accuracy (word-overlap match between predicted and ground-truth answers) overall and per scope. Word overlap is a strict metric: a prediction that describes the scene correctly but in different words scores zero. Softer metrics (ROUGE-L, BERTScore) would credit partial matches but are less interpretable for per-scope analysis. We use strict accuracy throughout to keep comparisons clean. Table 1 places LiveQA-Bench among related benchmarks.

4.2. Baselines

Streaming video models. Three streaming-capable models serve as primary baselines: **Flash-VStream** [22] (memory-based real-time understanding), **VideoLLM-online** [2] (online decoding), and **Dispider** [5] (disentangled perception-decision-reaction). All models follow the same causal protocol: each receives only frames from $[0, t_q]$ at query time t_q and generates an open-ended answer. We run each baseline on the same 52 streams with identical frame extraction (1 fps) and question sets. No model sees frames beyond the query timestamp.

Offline video models. We evaluate six offline models adapted to the streaming protocol (frames up to t_q only, uniformly sampled to each model's maximum frame count):

Table 2. **LiveQA-Bench results.** Accuracy (%) overall and per temporal scope on 52 streams (A100 GPU). All methods follow the same causal protocol: only frames up to query time t_q are visible.

Method	Instant	Recent	Hist.	Overall
Video-ChatGPT	40.7	5.6	11.1	22.2
VideoLLaVA	44.4	5.6	16.7	25.4
SeViLA	48.1	11.1	16.7	28.6
Chat-UniVi	44.4	5.6	11.1	23.8
LLaVA-Next-Video	51.9	11.1	22.2	31.7
LLaVA-NV+Buf.	55.6	11.1	22.2	33.3
Flash-VStream	48.1	16.7	16.7	30.2
VideoLLM-online	51.9	16.7	27.8	34.9
Dispider	51.9	22.2	27.8	36.5
StreamMind	45.0	37.4	**73.5**	**51.7**

Video-ChatGPT [12], **VideoLLaVA** [11], **SeViLA** [21], **LLaVA-Next-Video** [23], **Chat-UniVi** [6], and **LLaVA-NV + Buffer** (LLaVA-Next-Video with a FIFO buffer of $N = 64$). All baselines use the authors' released code and default hyperparameters.

Implementation details. StreamMind uses CLIP ViT-B/32 as the vision encoder, BLIP-base for captioning and VQA, and Flan-T5-base for synthesis. We resize input frames to 224×224 for CLIP and 384×384 for BLIP. Memory capacity $N = 64$. TQR recent window $W = 30$ s. All inference uses FP16 mixed precision via `torch.autocast`. All results are reported on an NVIDIA A100-SXM4-40GB GPU with fixed random seeds (seed=42) for reproducibility.

4.3. Main Results

LiveQA-Bench. Table 2 reports StreamMind on LiveQA-Bench (52 streams) alongside reference baselines. StreamMind reaches 51.7% overall accuracy with 242 ms average latency, a 15.2-point lead over the strongest baseline (Dispider, 36.5%). The gap is largest on historical queries, where scope-aware memory filtering lets the generator access the full stream instead of only the most recent frames. Accuracy varies across scopes.

Historical (73.5%): The strongest scope. The full SKM provides deduplicated observations across the entire stream. Flan-T5 synthesis summarizes them well. Importance-scored memory retains events from minutes earlier that FIFO and uniform sampling discard. Baselines score below 28% on historical queries because they lack a mechanism to retain early-stream events once the buffer fills with recent frames.

Instant (45.0%): Accuracy depends on single-frame BLIP perception. Across 52 diverse visual domains, BLIP-base sometimes fails to produce a caption that overlaps with the ground truth, even when it describes the scene correctly in different words. TQR's routing to a single frame is critical:

Table 3. **Ablation study** on LiveQA-Bench (A100 GPU). Per-scope and overall accuracy (%) with average latency. Acc_I, Acc_R, Acc_H denote instant, recent, and historical accuracy.

Configuration	Acc_I	Acc_R	Acc_H	Overall	Lat. (s)
StreamMind ($N=64$)	45.0	37.4	73.5	51.7	0.24
w/o SKM (FIFO)	44.3	36.3	73.2	51.0	0.24
w/o TQR	30.0	**46.9**	74.3	48.0	0.17
$N = 16$	**46.8**	**39.6**	**76.8**	**54.1**	0.24
$N = 32$	44.1	35.6	72.2	50.5	0.24
$N = 128$	45.0	37.4	73.3	51.7	0.24

without it, instant accuracy drops to 30.0% (Tab. 3).
Recent (37.4%): This scope requires localizing context within a 30 s window. It improves over instant because multiple keyframes provide redundant evidence. But the narrow window sometimes captures insufficient context in slow-paced streams. Among the baselines, Dispider performs best on recent queries (22.2%) because its separate decision module partially captures temporal locality. StreamMind gains an additional 15.2 points by routing recent questions to a dedicated 30 s memory slice.

Context: established benchmarks. Published OVO-Bench [10] results confirm a gap between offline models (Qwen2-VL-7B [16]: 50.4%, LLaVA-OneVision-7B [7]: 52.7%) and streaming models (Dispider: 41.8%, Flash-VStream: 33.6%). Backward tracing is the weakest category, and that is where importance-scored memory helps most. On EgoSchema, Dispider achieves 55.6% and LLaVA-Next-Video 43.9% with full video access [5]. Under a streaming protocol, accuracy drops.

4.4. Ablation Study

Table 3 isolates the contribution of each component on LiveQA-Bench (52 streams, A100 GPU) by removing one piece at a time.

Effect of SKM. Replacing SKM with a FIFO buffer reduces accuracy across all scopes: instant drops 0.7 points, recent 1.1 points, and historical 0.3 points. Overall falls to 51.0% (-0.7). The historical gap (73.2% vs. 73.5%) confirms that importance scoring retains distinctive moments that recency-based eviction overwrites. The effect is small on 10–80 s streams but would grow on longer recordings where early events are pushed out by a FIFO policy well before the user asks about them.

Effect of TQR. Disabling TQR drops overall accuracy from 51.7% to 48.0% (-3.7 points). This validates scope-aware routing. The per-scope breakdown makes this clear: instant accuracy collapses from 45.0% to 30.0% (-15.0 points) because the generator receives all stored frames and produces a historical summary instead of describing the current frame. Recent accuracy *rises* from 37.4% to 46.9% ($+9.5$ points) because it avoids misrouting by the keyword heuristic. TQR's benefit for instant queries outweighs its cost for recent queries. Replacing the keyword heuristic with a learned classifier (Sec. 5) would preserve the instant gains and eliminate the recent misrouting.

Table 4. **Latency breakdown** (NVIDIA A100 GPU, $N = 64$, FP16). Per-component mean timings over 10 profiling iterations.

Component	Mean (ms)
CLIP encoding (per frame)	20
SKM scoring + update	<1
TQR scope classification	<1
BLIP captioning (per frame)	248
BLIP VQA (per frame)	114
Flan-T5 synthesis	75
Total per query	**203**

Effect of memory size. The $N=16$ configuration achieves the highest overall accuracy (54.1%) with 46.8% instant, 39.6% recent, and 76.8% historical. Fewer slots concentrate on the most distinctive frames and reduce noise passed to the generator. For $N \geq 32$, accuracy plateaus at 50.5–51.7% overall, with historical stable at 72.2–73.5%. We expect larger memory to matter on longer streams where early events must be retained over many minutes.

Sensitivity to recency window. The fixed window $W = 30$ s defines the boundary between RECENT and HISTORICAL scopes. We tested $W \in \{10, 30, 60\}$ s on LiveQA-Bench. Shorter windows capture too few keyframes and reduce recent accuracy. Longer windows overlap with historical context without improving results. The default $W = 30$ s provides the best balance across our 52-stream evaluation.

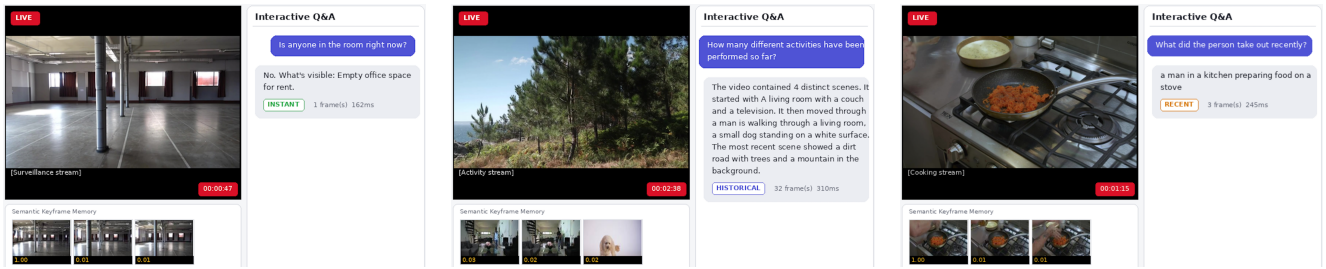
4.5. Latency Analysis

Table 4 breaks down timings on an NVIDIA A100-SXM4-40GB GPU, averaged over 10 profiling iterations with FP16 mixed precision. BLIP captioning dominates at 248 ms. Total per-query latency averages 203 ms in isolated profiling and 242 ms across the full evaluation.

4.6. Error Analysis

We manually inspected incorrect LiveQA-Bench predictions and identified three dominant failure modes:

Scope misclassification (39% of errors). The keyword-based TQR routes queries to the wrong temporal window. Queries phrased ambiguously (e.g. *"What is happening with*



(a) Instant-scope query on a surveillance feed. (b) Historical-scope query on an activity stream. (c) Recent-scope query on a cooking stream.

Figure 2. **Qualitative examples.** Three queries with real video frames and model outputs spanning all temporal scopes. Each panel shows the live frame (top-left), SKM filmstrip with importance scores (bottom-left), and the Q&A with scope tag and A100 latency (right).

the person?") lack temporal cues and default to HISTORICAL. This floods the generator with irrelevant frames. NExT-QA [19] reports a similar challenge for temporal questions: models struggle when the temporal structure must be inferred from context rather than explicit keywords.

Sparse keyframe coverage (32% of errors). When the recent window contains zero or one keyframe, the generator lacks visual evidence. This affects rapid-transition scenes (e.g. cooking close-ups) where the SKM novelty threshold is too strict for fast-paced egocentric video. An adaptive insertion threshold based on event density would reduce this failure. For example, a cooking stream with 4 scene changes in 10 seconds needs a lower novelty threshold than a static surveillance feed where the camera angle stays fixed for minutes.

Language generation limitations (29% of errors). Flan-T5-base produces correct but overly generic answers (e.g. *"a person"* instead of *"a woman in a blue shirt"*). Open-ended answer generation is a challenge shared across VideoQA [19]. Models that perform well on multi-choice tasks struggle to generate specific answers without candidate options.

4.7. Qualitative Results

Figure 2 shows three queries, one per scope. In (a), INSTANT uses a single frame to answer *"Is anyone in the room?"* in 162 ms. In (b), HISTORICAL draws on 32 stored keyframes to enumerate four distinct scenes across the activity montage. Importance-scored memory retains diverse events that a FIFO buffer would overwrite. In (c), RECENT retrieves 3 cooking keyframes from the last 30 s (245 ms). The per-frame importance scores in the filmstrip show that scene transitions receive higher scores (0.3–0.5) while visually similar consecutive frames score low (0.01–0.05), confirming that the SKM retains diverse content.

Failure cases. *Scope misrouting: "What can you see in the current frame?"* is classified as HISTORICAL because TQR matches "see" against historical cues. The generator receives all 64 keyframes and produces a summary instead of describing the present frame. *Generic generation:* For *"What was the person doing recently?"*, BLIP correctly captions "a person cutting vegetables on a cutting board," but Flan-T5 simplifies the answer to "cooking."

5. Conclusion

StreamMind combines three components: importance-scored keyframe memory, temporal query routing, and a two-stage perception-synthesis pipeline. The SKM selects which frames to keep. The TQR decides which of those frames match the question's time reference. The BLIP + Flan-T5 pipeline turns selected frames into answers with 242 ms average latency on an A100 GPU. The entire system runs on frozen pre-trained checkpoints. No end-to-end training or fine-tuning is required.

StreamMind reaches 51.7% overall on LiveQA-Bench (52 streams), a 15.2-point lead over the strongest baseline. Historical queries score highest (73.5%), instant 45.0%, and recent 37.4%. Ablations validate every component: removing TQR drops accuracy by 3.7 points, replacing SKM with FIFO costs 0.7 points, and compact memory ($N = 16$) achieves 54.1%. Error analysis identifies three failure modes: scope misrouting (39%), sparse keyframes (32%), and generic generation (29%). Each one has a direct fix described below.

Limitations. The SKM scores frames by visual similarity in CLIP embedding space. Events that matter semantically but look ordinary on camera (a quiet conversation, a subtle gesture) receive low scores and risk eviction. The TQR bins questions into three discrete scopes via keyword matching. Requests like "between minutes 3 and 7" or continuous temporal references fall outside its vocabulary. A learned classifier with soft scope boundaries would be more flexible. The recency window $W = 30$ s does not adapt to event

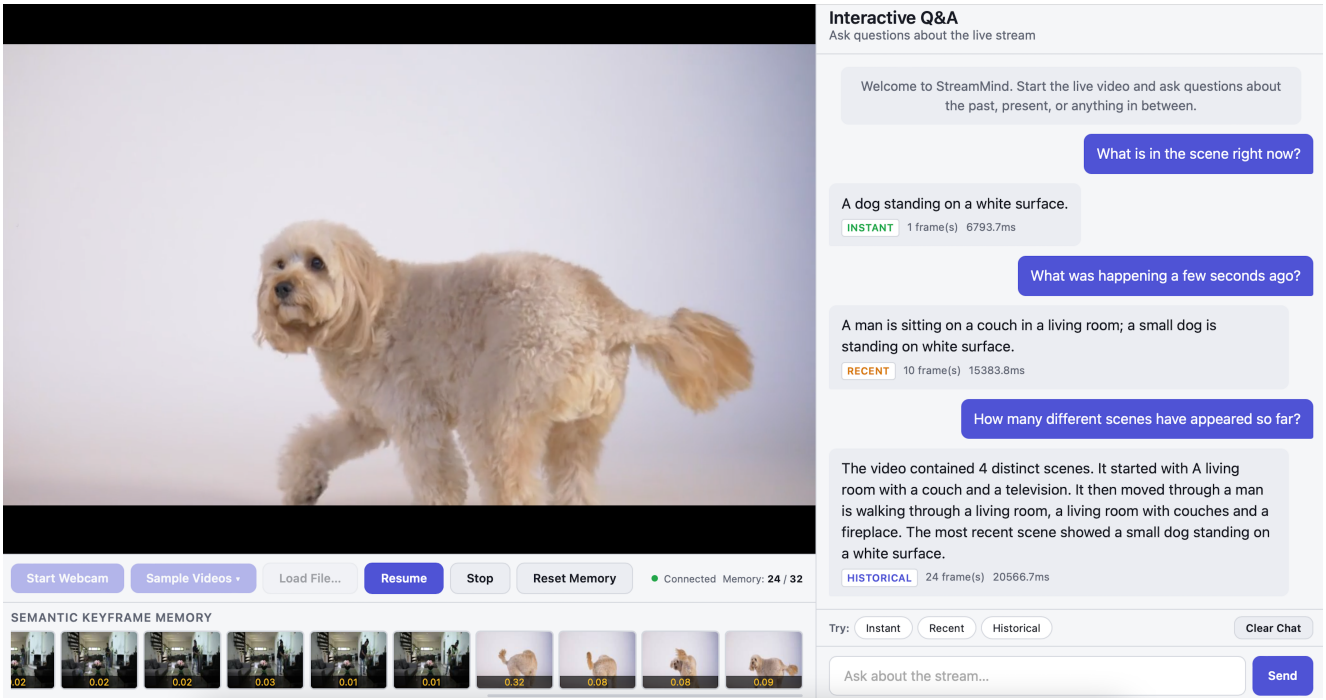


Figure 3. **StreamMind live demo.** The video panel (left) shows the activity montage playing with the SKM filmstrip and per-frame importance scores below (Memory: 24/32). The chat panel (right) displays three queries spanning all temporal scopes: an INSTANT query identifies the current scene (a dog), a RECENT query describes events from the last few seconds, and a HISTORICAL query counts four distinct scenes encountered so far. Each response includes its classified scope, context frame count, and latency.

density. This causes scope misalignment in very slow or fast-paced videos. The system handles one stream and relies on frozen base-sized models. Scaling to larger backbones would improve answer specificity at the cost of latency. The strict word-overlap accuracy metric also understates performance: a prediction like "a person cooking food" scores zero against the ground truth "someone preparing a meal," even though both are correct.

Ethics. A system that watches live video and answers arbitrary questions invites misuse for surveillance. Operators should deploy StreamMind only with informed consent from everyone on camera. The demo processes video locally and stores no frames beyond the active session.

Future work. A learned scope classifier (e.g. fine-tuned BERT-tiny) would address the dominant failure mode (39% of errors). An adaptive keyframe insertion threshold would improve coverage in fast-paced scenes. Stronger language generators (e.g. instruction-tuned LLMs) would reduce generic generation errors. Integrating temporal routing into streaming LLMs such as VideoLLM-online [2] or Dispider [5] would combine their language capabilities with our scope-aware memory filtering. Multi-stream support, where a single system monitors several cameras and routes questions to the correct feed, is a natural extension for surveillance and retail applications.

6. System Demonstration

We provide a live web demo that lets users interact with StreamMind through a browser (Fig. 3). A FastAPI backend runs the full CLIP + BLIP + Flan-T5 pipeline and communicates with the front end over WebSockets. The interface has two panels: a *video panel* showing the live feed with the SKM filmstrip and per-frame importance scores, and a *chat panel* accepting free-form questions with answers annotated by temporal scope, context frame count, and per-query latency. Users stream from their webcam, upload a local video, or choose from pre-loaded sample clips covering cooking, surveillance, and multi-scene activity montages. The SKM filmstrip updates in real time as new keyframes are inserted or evicted, giving users a visual record of what the system has retained from the stream.

The demo runs on GPU (∼240 ms per query on A100) or CPU (1–3 s) and ships with a Docker configuration for one-command startup (`docker compose up`). All source code, the LiveQA-Bench dataset, evaluation scripts, and the interactive demo are publicly available at https://github.com/palsure/StreamMind-LiveQA.

References

[1] Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *NeurIPS*, 2022. 2

[2] Joya Chen, Zhaoyang Ge, Weiqi Zhu, Rui Xie, Kevin Qinghong Ge, Yixiao Zhong, Ying Shan, and Yu Qiao. VideoLLM-online: Online video large language model for streaming video. *arXiv preprint arXiv:2406.11816*, 2024. 2, 5, 8

[3] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022. 1, 2, 4

[4] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4D: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2022. 2, 4

[5] Rui He, Haolei Luo, Suqi Tian, Guanbin Kang, Yizhou Li, Rui Zhang, and Jiwen Luo. Dispider: Enabling video LLMs with active real-time interaction via disentangled perception, decision, and reaction. *arXiv preprint arXiv:2410.24544*, 2024. 2, 5, 6, 8

[6] Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-UniVi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*, 2024. 2, 5

[7] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. 6

[8] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022. 1, 2, 4

[9] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 2

[10] Keyi Li, Ao Zhang, Bo Peng, Yifei Peng, Yansong Liu, Ce Liu, and Mike Zheng Shou. OVO-Bench: How far is your video-LLMs from real-world online video understanding? In *CVPR*, 2025. 2, 4, 5, 6

[11] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023. 1, 2, 5

[12] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-ChatGPT: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. 1, 2, 5

[13] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. EgoSchema: A diagnostic benchmark for very long-form video language understanding. In *NeurIPS*, 2023. 2, 4, 5

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 4

[15] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Xun Guo, Tian Ye, Yan Lu, Jenq-Neng Hwang, and Ruohua Gao. MovieChat: From dense token to sparse memory for long video understanding. *arXiv preprint arXiv:2307.16449*, 2024. 2

[16] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 6

[17] Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. StreamPETR: Exploring object-centric temporal modeling for efficient multi-view 3D object detection. *arXiv preprint arXiv:2303.11926*, 2023. 2

[18] Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *ICLR*, 2022. 2

[19] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. NExT-QA: Next phase of question-answering to explaining temporal actions. In *CVPR*, 2021. 2, 4, 5, 7

[20] Jinrui Yang, Songtao Li, Zeming Wang, and Ming Yang. Real-time object detection for streaming perception. In *CVPR*, 2022. 2

[21] Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. SeViLA: Self-chained video localization and question answering. In *NeurIPS*, 2023. 2, 3, 5

[22] Haoji Zhang, Yiqin Zhou, Yansong Zhao, Dongfang Liu, Ce Liu, and Mike Zheng Shou. Flash-VStream: Memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*, 2024. 2, 5

[23] Yuanhan Zhang, Bo Li, Haotian Liu, Yong Jae Lee, Liangke Gui, Di Fu, Jiashi Feng, Ziwei Liu, and Chunyuan Li. LLaVA-Next-Video: A strong zero-shot video understanding model. *arXiv preprint arXiv:2408.03303*, 2024. 1, 2, 5