AURA: ALWAYS-ON UNDERSTANDING AND REAL-TIME ASSISTANCE VIA VIDEO STREAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Video Large Language Models (VideoLLMs) have achieved strong performance on many video understanding tasks, but most existing systems remain offline and are not well-suited for live video streams that require continuous observation and timely response. Recent streaming VideoLLMs have made progress, yet current approaches often rely on decoupled trigger-response pipelines or are limited to captioning-style narration, reducing their effectiveness for open-ended question answering and long-horizon interaction. We propose **AURA** (**A**lways-On **U**nderstanding and **R**eal-Time **A**ssistance), an end-to-end streaming visual interaction framework that enables a unified VideoLLM to continuously process video streams and support both real-time question answering and proactive responses. AURA integrates context management, data construction, training objectives, and deployment optimization for stable long-horizon streaming interaction. It achieves state-of-the-art performance on streaming benchmarks and supports a real-time demo system with ASR and TTS running at 2 FPS on two 80 GB GPUs.

1 INTRODUCTION

Video Large Language Models (VideoLLMs) (Zhang et al., 2023; Maaz et al., 2024; Lin et al., 2024; Ren et al., 2024; Li et al., 2024a) have recently emerged as a core paradigm for video understanding by integrating visual perception with language-based reasoning and response. These models have demonstrated strong performance across a wide range of tasks, including video captioning (Yang et al., 2023; Wang et al., 2024; Abdar et al., 2024), video question answering (Li et al., 2024b; Wu et al., 2024; Fu et al., 2025a; Song et al., 2025), and temporal grounding (Lin et al., 2023; Qu et al., 2024). They also support diverse real-world applications, such as public safety monitoring (Cournan et al., 2016; Ahn et al., 2023) and intelligent transportation (Wan et al., 2020; Vishal et al., 2024; Zhang et al., 2025a). These advances highlight the potential of VideoLLMs to become a foundational technology for next-generation intelligent visual systems.

Despite this progress, most existing VideoLLMs are still designed for offline settings (Shen et al., 2024; Zhang et al., 2024; Wang et al., 2025a). In this paradigm, a complete video or a pre-collected segment is first buffered and then analyzed. Although this setup is well-suited for post hoc analysis, it limits the system's ability to respond promptly to ongoing events, making it less effective for applications such as real-time AI assistants (Qian et al., 2025; Wang et al., 2026a; Seed, 2026), live video narration (Chen et al., 2025; Xu et al., 2026), and interactive robotic systems (Lynch et al., 2023; Fang et al., 2025). To address this limitation, recent studies have begun to explore streaming VideoLLMs (Chen et al., 2024; Yao et al., 2025; Wang et al., 2026b; Seed, 2026). Systems built on streaming VideoLLMs can continuously process incoming video frames, incrementally update their understanding of the environment, and provide real-time responses. More importantly, they move beyond the traditional paradigm in which systems respond only after receiving explicit queries, enabling proactive responses based on information discovered from the video stream.

Recent research on streaming VideoLLMs has explored effective designs in architectural design, model training, and deployment efficiency (Chen et al., 2024; Xu et al., 2026; Qian et al., 2025; Wang et al., 2026b). Despite these advances, existing approaches still face significant challenges. Current streaming VideoLLMs mainly fall into two categories: decoupled architectures and unified architectures. Decoupled architectures (Qian et al., 2025; Wang et al., 2026a) rely on two separately deployed models, where a trigger model determines whether the primary VideoLLM should respond.

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

Because the trigger model does not share the same contextual state with the primary model and is typically much smaller in scale, the triggering accuracy and its consistency with response generation may be limited, leading to unstable system behavior. Unified architectures offer a higher performance upper bound, but these works (Chen et al., 2024; Xu et al., 2026) are generally limited to captioning-style narration tasks and remain less effective for complex open-ended video question answering. Although recent full-duplex models (Cui et al., 2026) integrate question answering into a single framework, they still lack sufficient robustness for long-duration streaming and may suffer from memory overflow or performance degradation during extended inference.

To address these issues, we introduce **AURA**, an end-to-end streaming visual interaction framework powered by a streaming VideoLLM. Through the co-design of data, algorithms, and systems, AURA achieves high stability, broad capabilities, and long-term endurance. Specifically, AURA tackles three challenges in streaming video understanding: (1) enabling a unified model to process video streams frame by frame and decide whether to remain silent or generate an appropriate response; (2) extending unified streaming models beyond captioning-style narration to support complex open-ended video question answering; and (3) stably handling unbounded video-text inputs over extended durations. To this end, we propose **Interactive Video Stream Context Management**, which integrates unbounded video frame streams and textual question-answer (QA) interactions into a limited context in an appropriate manner, enabling the model to perform silent observation and support diverse response modes. Based on the streaming response modes defined by this mechanism, we further develop a **Coarse-to-Fine Data Engine** with a five-stage pipeline to systematically construct training data for Real-Time QA, Proactive QA, and Multi-Response QA. In addition, to balance continuous silence with timely responses, we introduce a **Silent-Speech Balanced Loss** during training. Furthermore, we implement a **Real-Time Streaming Inference Framework**, leveraging KV-cache reuse and related optimizations to enable efficient real-time inference over unbounded video streams.

We develop AURA on top of Qwen3-VL-8B-Instruct (Bai et al., 2025) as a unified streaming visual assistant for real-time and real-world interaction over continuous video streams. After training, AURA achieves state-of-the-art results on streaming benchmarks while maintaining competitive performance on offline video understanding tasks. To further support practical real-time assistance, we integrate Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) into the system and develop a functional demo application. With deployment optimizations, the system runs in real time at 2 FPS on two 80 GB GPUs. The primary contributions of our work are summarized as follows:

(1) We study streaming visual interaction with VideoLLMs as a setting that requires continuous observation, selective silence, and timely response over unbounded video streams.
(2) We propose AURA, an end-to-end streaming visual interaction framework that empowers a unified VideoLLM to process continuous frames for both real-time QA and proactive response, ensuring timely system-level stability over unbounded video streams.
(3) We develop a co-designed pipeline spanning data construction, training, and deployment for stable long-horizon streaming, including Interactive Video Stream Context Management for context organization, a Coarse-to-Fine Data Engine for data construction, a Silent-Speech Balanced Loss for training, and a Real-Time Streaming Inference Framework for deployment.
(4) AURA achieves state-of-the-art results on streaming benchmarks while maintaining competitive performance on offline video understanding tasks. To demonstrate its practical usability, we build a real-time demo system with ASR and TTS, which runs at 2 FPS on two 80 GB GPUs.

2 Related Work

2.1 Video Large Language Models

Research on VideoLLMs has moved from simple feature alignment to more efficient token management and deeper reasoning. To balance spatial resolution and temporal coverage, Keye-VL-1.5 (Yang et al., 2025a) introduces a Slow-Fast video encoding strategy that dynamically allocates computation according to inter-frame similarity. Qwen3-VL (Bai et al., 2025) enhances spatiotemporal modeling through interleaved-MRoPE and a text-based temporal alignment mechanism. To reduce computational redundancy, InternVL3.5 (Wang et al., 2025a) adopts Visual Resolution Routing for patch-level visual token compression, while LongVU (Shen et al., 2024) combines DINOv2 (Oquab et al., 2023) features with cross-modal queries for adaptive spatiotemporal compression. Recent studies also explore higher-level temporal reasoning and event modeling. VideoChat-A1 (Wang et al., 2026c)

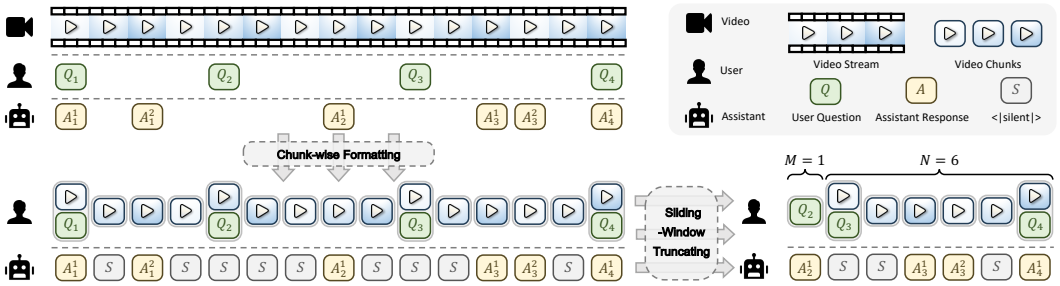


Figure 1: Overview of our Interactive Video Stream Context Management mechanism. The framework uses a dual sliding-window strategy, where N denotes the length of the video window, and M denotes the number of recent QA groups retained in the interaction history outside the video window.

introduces Chain-of-Shot reasoning for stepwise long-video understanding, and VidEvent (Liang et al., 2025) provides large-scale annotated data for dynamic event modeling. Despite these advances, most existing VideoLLMs are still designed primarily for offline settings.

2.2 Streaming Video Understanding

The core of streaming video understanding lies in the shift from whole-video processing to online interactive processing (Wang et al., 2026a; Xia et al., 2026), requiring synchronization of perception, decision-making, and response. Dispider (Qian et al., 2025) proposes a decoupled framework using a lightweight active streaming module to detect interaction points and avoid autoregressive decoding latency. MMDuet2 (Wang et al., 2026b) applies multi-round reinforcement learning and a PAUC reward to improve response quality and timeliness. MiniCPM-o-4.5 (Cui et al., 2026) extends omni-modal modeling to full-duplex live streaming with simultaneous video and audio input together with concurrent text and speech output. LiveCC (Chen et al., 2025) shows that low-cost ASR transcripts can serve as scalable supervision for streaming training. StreamingVLM (Xu et al., 2026) enables stable real-time understanding through compact cache-based streaming inference and chunk-level fine-tuning. VideoLLM-online (Chen et al., 2024) introduces a LIVE framework combining streaming-oriented training, dialogue data generation, and efficient real-time inference. Eyes Wide Open (Zhang et al., 2025b) presents an ego-proactive VideoLLM for streaming video that focuses on proactive coherence and just-in-time responsiveness using multi-stage training and dynamic compression. These advances have been evaluated on OVO-Bench (Niu et al., 2025), StreamingBench (Lin et al., 2026), and OmniMMI (Wang et al., 2025b). Despite this progress, achieving a unified framework that jointly optimizes context management, data construction, training, and real-time deployment for stable long-horizon streaming interaction remains a significant challenge.

3 Interactive Video Stream Context Management and QA Types

Building continuous and timely visual interaction modeling for streaming scenarios is highly challenging. In streaming video understanding, both the input video and the interaction history between the user and the model (*i.e.*, the assistant) grow continuously over time, making the continuity unbounded. However, the context window of an LLM is fundamentally limited, and an excessively long context can slow inference, thereby compromising timeliness. To address this trade-off, AURA introduces an Interactive Video Stream Context Management mechanism (Sec. 3.1), as illustrated in Fig. 1. Based on this mechanism, three streaming QA modes are defined (Sec. 3.2).

3.1 Interactive Video Stream Context Management

Chunk-wise Conversational Format. During real-world streaming inference, the video stream is collected in small temporal chunks (*e.g.*, one chunk per second), and we therefore organize the context in a chunk-wise conversational format. For each video chunk, if a user question is issued at that time, the question and the corresponding video chunk are packaged together into a user message. Otherwise, the user message contains only the video chunk and no text. Each user message is followed by an assistant message. If there is an assistant response at that time step, the assistant message

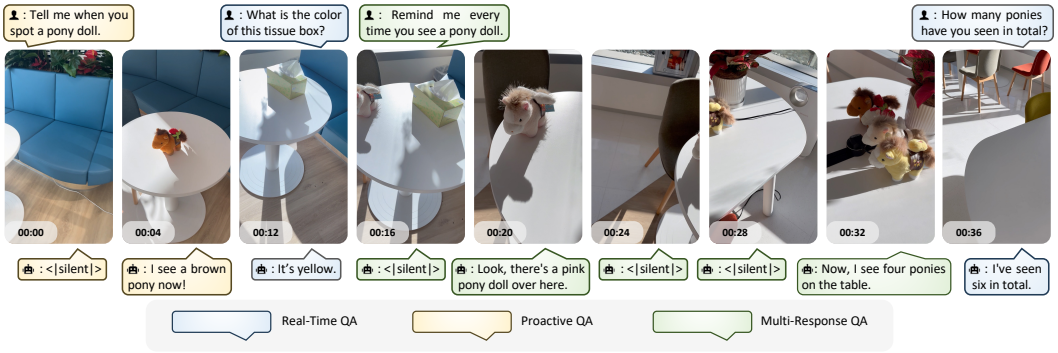


Figure 2: Streaming QA types. **Real-Time QA** produces a single immediate response at the query time. **Proactive QA** produces a single delayed response after sufficient future evidence is observed. **Multi-Response QA** continuously tracks evolving events and produces multiple responses over time.

contains the response text. Otherwise, it is filled with a special token, `<|silent|>`, indicating that the model remains silent at that moment.

Dual Sliding-Window Strategy. To manage the unbounded growth of the streaming video and interaction history, we adopt a dual sliding-window strategy over both the video stream and the QA interaction stream. For the video stream, we maintain a sliding window that keeps only the most recent N seconds of video. Because visual tokens are highly dense and user-relevant information is typically associated with recent visual content, N is set to a relatively small value (*e.g.*, $N = 30$). In contrast, QA interactions are text-based, token-efficient, and often carry critical user intent and historical context. Therefore, outside the video window, we maintain a separate sliding window over QA interactions that preserves the most recent M QA groups (*e.g.*, $M = 10$), where a QA group is defined as a user question together with all subsequent non-silent assistant responses. If the total video duration does not exceed N seconds, the full history is retained. Otherwise, the context is truncated jointly by the video and QA windows. For the M QA groups that fall outside the N-second video window, all video chunks and `<|silent|>` tokens are discarded, and only the non-silent textual content of the user and assistant messages is preserved. Overall, this dual sliding-window strategy preserves important historical information while controlling the total context length and enabling efficient computation.

3.2 Streaming QA Types

Based on the context management mechanism in Sec. 3.1, user-assistant interaction is no longer restricted to the traditional turn-taking paradigm. Instead, both the user and the assistant can interact asynchronously over a continuous video stream. In real-world streaming settings, user queries exhibit diverse temporal dependencies. Some can be resolved immediately, others require waiting for future evidence, and some pertain to ongoing events requiring continuous monitoring. Based on these interaction patterns, we categorize streaming QA interactions into three types:

(1) Real-Time QA, where the model produces a single immediate response grounded in the currently available or previously observed visual context.
(2) Proactive QA, where the model remains silent after receiving the query and generates a single response only after sufficient visual evidence has been accumulated.
(3) Multi-Response QA, where the model generates multiple responses over time as new visual information becomes available.

Examples of these interaction types are illustrated in Fig. 2. Together, these three categories form the foundation of our data construction pipeline (Sec. 4), enabling the model to learn diverse response behaviors under streaming conditions.

4 Coarse-to-Fine Streaming Data Engine

Building on the QA categories defined above, we develop a coarse-to-fine data engine to generate streaming training data for Real-Time QA, Proactive QA, and Multi-Response QA. As illustrated in Fig. 3, the pipeline consists of five stages: (1) **Video Preparation**, (2) **QA Synthesis**, (3) **QA Refine-**

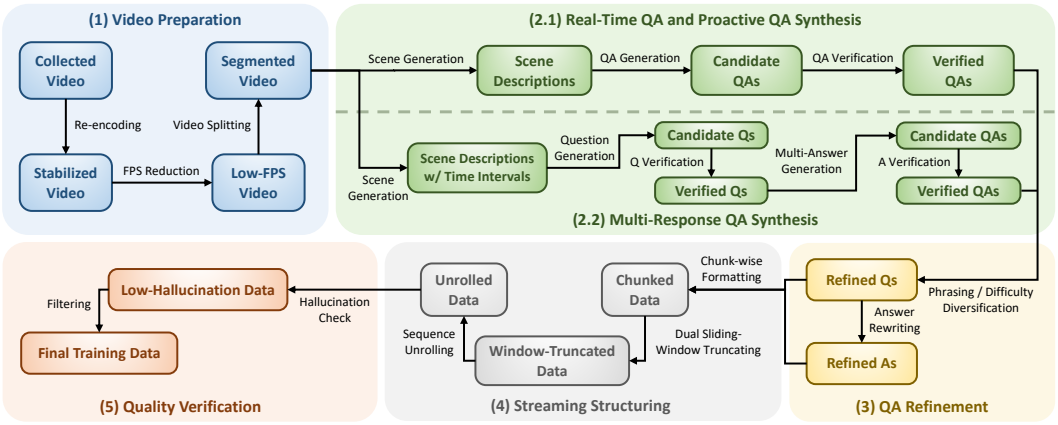


Figure 3: Overview of the Coarse-to-Fine Streaming Data Engine in AURA. The pipeline comprises five stages: (1) Video Preparation, (2) QA Synthesis, (3) QA Refinement, (4) Streaming Structuring, and (5) Quality Verification.

ment, (4) **Streaming Structuring**, and (5) **Quality Verification**. This pipeline translates the interaction taxonomy into structured supervision, enabling the model to learn both when to respond and what to respond under different interaction patterns, thereby providing a strong data foundation for AURA.

(1) Video Preparation. We collect high-quality videos from public internet sources covering diverse categories (e.g., sports, vlogs, documentaries, TV shows, games). Each video is resampled to 2 FPS and re-encoded in H.264 (Wiegand et al., 2003) for codec consistency.

(2) QA Synthesis. We use two pipelines to synthesize streaming QA because the three QA types have different temporal structures. Real-Time and Proactive QA require only one answer, while Multi-Response QA involves multiple answers over time. For Real-Time and Proactive QA, a multimodal large language model (MLLM) first performs scene segmentation and produces scene-level descriptions. It then generates candidate QA pairs with timestamps and verifies each pair against the video content observed up to the answer timestamp. For Multi-Response QA, we use a separate pipeline that first generates candidate questions with timestamps, then checks whether each question admits multiple valid answers at different timestamps, and finally produces and verifies those answers. Only QA pairs that pass verification are retained. Full details are provided in Appendix A.1.

(3) QA Refinement. We find the QA samples from the previous stage exhibit limited diversity. Specifically, Real-Time QA lacks diversity in difficulty levels, while Proactive and Multi-Response QA lack diversity in question phrasing. To address this issue, we augment each Real-Time QA timestamp with four additional questions covering increasing difficulty levels and sample one for the final pair. For Proactive and Multi-Response QA, we rewrite questions into semantically equivalent forms with predefined templates. Details are in Appendix A.2.

(4) Streaming Structuring. This stage converts timestamped QA annotations into training samples that match the Interactive Video Stream Context Management mechanism. In actual streaming inference, the model generates each response online based on the video window and the retained QA history available at that moment. Accordingly, during training, answers with different timestamps within the same continuous QA sequence should be paired with their corresponding video content and contextual history. We therefore unroll each sequence of continuous QA interactions from the same video into multiple training samples, each containing the interaction history up to one non-silent assistant message to be supervised, which we refer to as the target answer. For each sample, we use the timestamp of the target answer as the anchor for determining the time window. During training, we supervise only this target answer, while retaining the earlier QA interactions as contextual history (the specific supervision target is described in Sec. 5.1). This design helps bridge the gap between the necessary window truncation in long streaming contexts and the need for reliable supervision signals.

(5) Quality Verification. Since the previous stage reformats each sample within a truncated streaming context window, the retained video content and contextual history may become insufficient to support the target answer, increasing the risk of hallucination. We therefore introduce a Quality Verification stage that checks whether each target answer is visually grounded, factually correct, and temporally consistent given the retained context. Only samples passing verification are retained. Type-specific verification criteria are detailed in Appendix A.3.

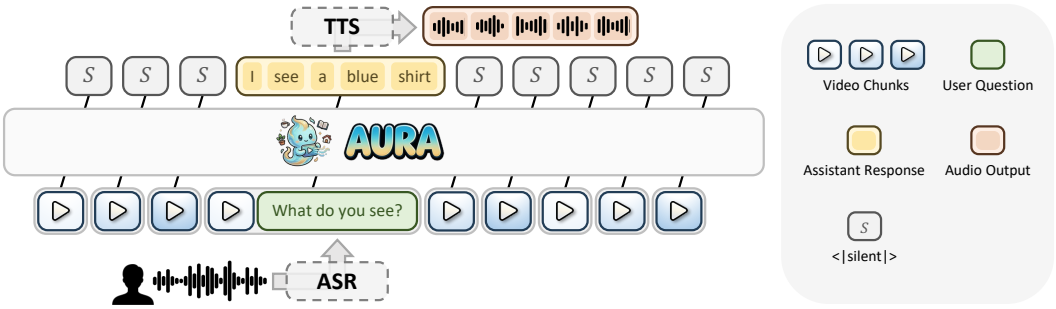


Figure 4: Overview of AURA's end-to-end real-time inference system with video and speech input, multimodal inference, and speech output. The system is designed to support continual streaming perception and low-latency interaction.

Additional Details. Beyond the five-stage pipeline, we introduce two practical designs. For Proactive and Multi-Response QA, a short acknowledgment is inserted after each user query to indicate receipt. We also merge QA samples of all three types from the same video into mixed interaction sequences ordered by timestamp, helping the model handle interleaved patterns in realistic streaming scenarios.

5 Training and Inference Design

To better address the unique challenges of streaming video understanding, we design both an effective training objective and an efficient inference framework for AURA. We introduce a loss that supports appropriate supervision selection and mitigates silent-turn bias in streaming interactions (Sec. 5.1). We also develop a real-time inference framework (Sec. 5.2) that enables low-latency, long-horizon assistance over continuous video streams.

5.1 Silent-Speech Balanced Loss

Supervision Target. Due to the sliding-window construction of the training data, a challenge arises in deciding which assistant messages should be supervised. Because each training sample is truncated by the windowing process, not every non-silent assistant message remains fully supported by the retained visual and history evidence. As described in Sec. 4, during data construction, we can guarantee only that the last non-silent assistant message in each sample is paired with the video window and contextual history anchored to its timestamp, and our quality verification in Sec. 4 is also performed with respect to this target answer. Earlier responses may have been correct in the original full sequence, but after truncation, the remaining context evidence may no longer be sufficient to justify them. Directly using all the non-silent assistant messages as supervision targets would encourage the model to produce responses from incomplete evidence, thereby increasing the risk of hallucination. We therefore apply loss only to all silent assistant messages and the last non-silent assistant message in each training sample, while excluding earlier non-silent assistant messages.

Class-Balancing Loss Strategy. Another major challenge is the imbalance between silence and speaking. In streaming interactions, the assistant should remain silent at most timestamps and respond only when the video context or the user query truly calls for it. Consequently, in streaming training data, the special token `<|silent|>` appears much more frequently than non-silent assistant messages. Moreover, because we supervise only the last non-silent assistant message, this imbalance is further amplified in the supervision signal. If all assistant messages were optimized with a standard cross-entropy loss, the model could become overly biased toward predicting silence. To address this issue, we adopt a class-balancing strategy for assistant-side supervision. Let N_{silent} denote the number of silent assistant messages in a training sample. We assign weight 1 to target tokens from non-silent responses and down-weight target tokens from silent assistant messages by the inverse imbalance ratio, *i.e.*, $w_{\text{silent}} = \frac{1}{N_{\text{silent}}}$, so that the two types of supervision contribute at a comparable scale. This strategy preserves the abundant silent supervision needed to learn when not to speak, while preventing it from dominating the optimization signal for informative assistant responses.

Combining supervision selection and class reweighting, the final training objective is

$$\mathcal{L} = -\frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \, w_t \log p_\theta(y_t \mid x, y_{<t}), \quad w_t = \begin{cases} \dfrac{1}{N_{\text{silent}}}, & y_t \text{ belongs to a silent message,} \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

where T is the total number of target tokens in the current training instance, y_t is the target token at position t, and x denotes the multimodal streaming input. Here $m_t \in \{0, 1\}$ is a supervision mask: $m_t = 1$ if y_t belongs to a silent assistant message or to the last non-silent assistant message in the instance, and $m_t = 0$ otherwise.

5.2 Real-Time Streaming Inference Framework

To support AURA for continuous streaming perception and real-time audio-video interaction, we develop an end-to-end inference system, as illustrated in Fig. 4. The system is centered on the AURA model, integrates ASR and TTS modules, and incorporates several efficiency optimizations.

Overall Inference Pipeline. The overall context is constructed following the Interactive Video Stream Context Management mechanism in Sec. 3.1. On the input side, the video stream and user speech are captured simultaneously. The video stream is divided into small chunks according to a predefined chunk size. When no user speech is detected, each video chunk is inserted into the context as an independent user message. When speech is received, the ASR module first transcribes it into text, and the transcript is then combined with the concurrent video chunk to form a single user message. Each new user message triggers the AURA model to generate a response. If the response is non-silent, it is sent to the streaming TTS module and delivered as speech. The generated output is also written back to the context as an assistant message. Throughout the system, AURA, ASR, and TTS operate asynchronously. We further incorporate streaming output, computation graph optimization, and multi-process resource isolation to support stable real-time interaction.

Prefix Caching Strategy. During deployment, context truncation is required once the total context reaches the predefined video window size N. A common approach is to maintain all video chunks in the context as a fixed-length first-in-first-out (FIFO) queue, but this makes the context prefix change at every step and prevents KV-cache reuse. To address this issue, we adopt a batch removal strategy that allows the video window to slightly extend within a margin of N'. When the window grows to $N + N'$, we remove the oldest N' video chunks together with the corresponding silent assistant messages. During the following N' insertions, no further removal is performed, so the prefix remains unchanged and prefix caching can be reused continuously. Only when the window again reaches $N + N'$ do we truncate it back to N and recompute the KV-cache for the remaining context. For the text QA history, we do not maintain a separate truncation trigger. Instead, whenever the video window is truncated, we also trim the QA groups outside the video window back to M. This design significantly avoids extra cache recomputation and improves inference speed. The detailed truncation schedule is given in Appendix B.1.

6 Experiment

We evaluate AURA in terms of benchmark accuracy, inference efficiency, and training design. We first present the implementation details and evaluation protocol (Secs. 6.1 and 6.2), then report the main benchmark and inference results (Secs. 6.3 and 6.4), and finally study offline capability and the effect of the Silent-Speech Balanced Loss (Sec. 6.5).

6.1 Implementation Details

All experiments are conducted on a computing cluster equipped with high-performance GPUs with 80 GB of HBM3 memory. Training is performed on 4 nodes with 8 GPUs per node (32 GPUs in total). We initialize our model from Qwen3-VL-8B-Instruct (Bai et al., 2025) and fine-tune only the LLM component while keeping the vision encoder and the connector frozen. The training data include approximately 115k streaming video QA samples (about 1.04B tokens), constructed using the data pipeline described in Sec. 4, as well as approximately 59k in-house offline video QA samples (about 0.16B tokens). In total, the training set contains approximately 174k samples, corresponding to about

Table 1: Quantitative comparison on StreamingBench in terms of accuracy (%). We report performance on each fine-grained task, along with the average results for Real-Time Visual Understanding (RTVU), Omni-Source Understanding (OSU), Contextual Understanding (CU), and the overall average. The best and second-best results are highlighted in bold and underlined, respectively.

Method	RTVU											OSU					CU					All
	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	Avg	ER	SCU	SD	MA	Avg	MCU	ACU	SQA	PO	Avg	
Proprietary Models																						
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	60.2
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
Open-Source Models																						
StreamAgent	79.6	78.3	79.3	75.9	74.7	76.9	82.9	66.3	73.7	55.4	74.3	35.9	26.3	38.9	44.0	36.3	39.7	30.3	39.6	28.9	34.6	57.0
ViSpeak	79.8	71.1	81.4	78.8	74.5	70.1	63.9	64.2	71.4	28.0	70.4	47.2	**56.4**	61.6	**81.2**	61.6	49.2	36.4	39.2	50.8	43.9	62.6
Qwen3-VL-8B-Inst.	80.1	76.6	81.1	81.1	77.0	73.2	80.6	63.4	68.8	**57.0**	74.1	44.4	27.2	46.4	49.6	41.9	30.0	38.4	38.0	52.8	39.8	59.3
MiniCPM-o-4.5	80.9	**88.3**	82.0	85.0	78.9	85.4	78.7	67.1	75.6	56.0	78.2	44.4	28.0	41.6	54.4	42.1	35.2	44.0	51.2	47.6	44.5	62.7
AURA (Ours)	**87.5**	84.4	**93.4**	**89.5**	**83.2**	**87.9**	**86.1**	**76.4**	**82.4**	47.7	**83.2**	**56.4**	48.4	**62.8**	80.4	**62.0**	**54.8**	**70.8**	**57.2**	**53.2**	**59.0**	**73.1**

1.2B tokens. Detailed data distributions are shown in Appendix A.4. The model is trained for one epoch with a global batch size of 128 and a learning rate of 1×10^{-5}. For context management, the video chunk size is set to 1 second, and the hyperparameters N, N', and M are set to 30, 15, and 10.

6.2 Evaluation Protocol

Benchmarks. We evaluate AURA on three streaming video understanding benchmarks. StreamingBench (Lin et al., 2026) focuses on real-time visual, omni-source, and contextual understanding. OVO-Bench (Niu et al., 2025) emphasizes temporal awareness under backward tracing, real-time understanding, and forward active responding. OmniMMI (Wang et al., 2025b) evaluates streaming video understanding, proactive reasoning, and multimodal interaction.

Evaluation Pipeline. We conduct all evaluations using official benchmark codebases (Lin et al., 2026; Niu et al., 2025; Wang et al., 2025b) and manage our model context with the Interactive Video Stream Context Management mechanism. For baselines, we report official results when available (Fu et al., 2025b; Wang et al., 2025b; Xia et al., 2026; Yang et al., 2025b; Zhang et al., 2025b). Otherwise, we evaluate them following official benchmark code (Cui et al., 2026; Bai et al., 2025). We note that MiniCPM-o-4.5's full-duplex live-streaming mode often becomes silent in this setting and may produce irrelevant responses for streams longer than two minutes, so we do not use this mode in our evaluation.

6.3 Main Results

We compare AURA with strong proprietary baselines, including GPT-4o (Hurst et al., 2024) and Gemini-1.5-Pro (Team et al., 2024), as well as open-source baselines, including StreamAgent (Yang et al., 2025b), Streamo-7B (Xia et al., 2026), ViSpeak (Fu et al., 2025b), VideoLLM-EyeWO (Zhang et al., 2025b), M4 (Wang et al., 2025b), Qwen3-VL-8B-Instruct (Bai et al., 2025), and MiniCPM-o-4.5 (Cui et al., 2026), on three representative streaming video benchmarks: StreamingBench (Lin et al., 2026), OVO-Bench (Niu et al., 2025), and OmniMMI (Wang et al., 2025b).

Accuracy Comparison. As reported in Tab. 1, AURA achieves the highest overall accuracy of 73.1% on StreamingBench (Lin et al., 2026), outperforming the strongest open-source baseline, MiniCPM-o-4.5, by 10.4%. This advantage is broad rather than concentrated on a few tasks. AURA ranks first among open-source models across all three high-level groups, *i.e.*, Real-Time Visual Understanding (RTVU), Omni-Source Understanding (OSU), and Contextual Understanding (CU), and on 14 of the 18 fine-grained sub-tasks. AURA is also highly competitive with proprietary models, surpassing Gemini-1.5-Pro by 6.0% overall (73.1% vs. 67.1%), while leading all proprietary models on the three group averages and on 15 of the 18 fine-grained sub-tasks.

AURA also achieves the best overall accuracy on OVO-Bench (65.3%) and OmniMMI (25.4%); detailed results and analyses are provided in Appendix C.1 and Appendix C.2, respectively.

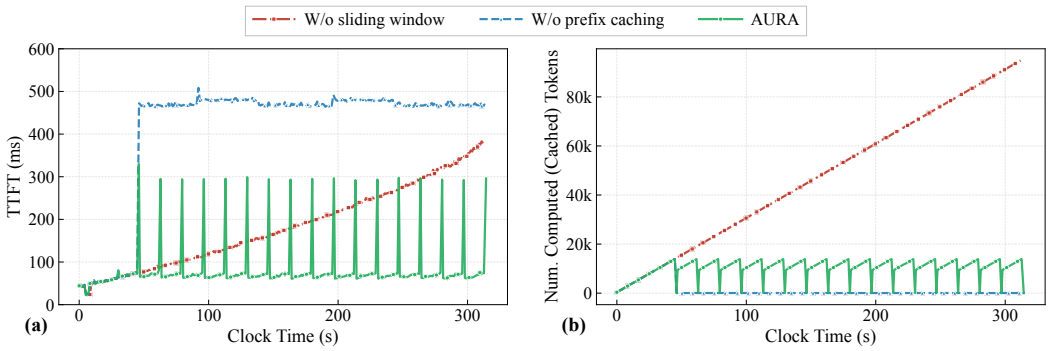


Figure 5: Inference performance of AURA. The figure compares (a) TTFT and (b) computed-token count across three settings: *w/o sliding window*, *w/o prefix caching*, and *AURA*.

6.4 Inference Performance

TTFT of AURA Model. We implement the AURA inference engine on top of vLLM (Kwon et al., 2023) on the same high-performance GPUs described in Sec. 6.1. We compare three configurations: *w/o sliding window*, *w/o prefix caching* ($N' = 1$), and *AURA* ($N' = 15$). We report time-to-first-token (TTFT), defined as the server-side latency between issuing a user query and receiving the first generated text token. As shown in Fig. 5, *AURA* achieves the lowest TTFT. Disabling prefix caching keeps TTFT consistently high due to repeated recomputation, while removing the sliding window causes latency to grow as context accumulates. This trend is corroborated by the right panel. Without sliding-window pruning, the amount of active cached computation grows steadily throughout the stream, while *AURA* keeps it bounded and sustains lower TTFT. This result shows that both sliding-window context management and prefix cache reuse are essential for real-time responsiveness.

End-to-end Latency. We also conduct end-to-end latency analysis for the whole system. The latency from the user's speech input to the first spoken response is approximately 312.2 ms, comprising ASR (84.2 ms), TTFT (75.0 ms), first-sentence decoding ($\sim$60 ms), and TTS first-chunk generation (93.0 ms). This supports real-time conversational interaction. TTS operates in a sentence-level streaming fashion and begins synthesis once the first sentence ($\sim$8 tokens) is decoded; the remaining tokens are decoded in parallel and do not add to user-perceived latency. Detailed experimental setup and per-component analysis are provided in Appendix C.3.

6.5 Research Questions

We further investigate two research questions in Appendix C.4: (1) whether streaming-oriented training preserves offline video understanding capability, and (2) the effect of our Silent-Speech Balanced Loss. For RQ1, AURA achieves 58.8% on LongVideoBench, 68.1% on MVBench, and 65.1% on Video-MME, remaining close to the base model Qwen3-VL-8B-Instruct and indicating that streaming training largely preserves offline ability (Tab. 4). For RQ2, replacing our objective with the default cross-entropy loss drops overall OmniMMI accuracy from 25.4% to 16.4% and Proactive Alerting from 37.5% to 0.0%, confirming that the Silent-Speech Balanced Loss is essential for preventing excessive silence and reducing hallucination (Tab. 5).

7 Conclusion

We present **AURA**, an end-to-end streaming framework that enables a unified VideoLLM to continuously observe live video streams, remain selectively silent, and provide both real-time and proactive responses. Through the co-design of data construction, model training, and system deployment, AURA achieves state-of-the-art performance on streaming benchmarks while maintaining competitive offline video understanding capability, and runs in real time at 2 FPS on two 80 GB GPUs with ASR and TTS integration. These capabilities may enable more timely and context-aware visual assistance in real-world streaming settings, with potential benefits for everyday human activities.

AI USE STATEMENT

Generative AI tools were used in the construction of our synthetic streaming video QA data, including candidate generation, question rewriting, and model-based quality verification, as described in Sec. 4 and Appendix A. They were also used to assist with drafting and improving the readability of parts of the manuscript. AI-generated data were filtered by our multi-stage verification pipeline, and all AI-assisted manuscript content, experimental results, and claims were reviewed by the authors. The authors take full responsibility for the final content of this work and all artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve a new human-subject study. Our data engine operates on publicly accessible videos; any release will redistribute only content for which redistribution is permitted and will otherwise provide identifiers, metadata, and processing code. Because always-on video assistants may capture sensitive information or be misused for surveillance, deployment should require informed user consent, visible recording indicators, access controls, and appropriate data-retention policies. The model may also inherit demographic, cultural, and domain biases from its base model and training data and should not be relied upon for safety-critical decisions without human oversight.

REPRODUCIBILITY STATEMENT

We provide the data-construction procedure in Sec. 4 and Appendix A, the training configuration in Sec. 6.1, the evaluation protocol in Sec. 6.2, and inference and ablation settings in the appendix. The AURA model with a real-time inference framework will be made publicly available.

REFERENCES

Moloud Abdar, Meenakshi Kollati, Swaraja Kuraparthi, Farhad Pourpanah, Daniel McDuff, Mohammad Ghavamzadeh, Shuicheng Yan, Abduallah Mohamed, Abbas Khosravi, Erik Cambria, et al. A review of deep learning for video captioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Jungmo Ahn, JaeYeon Park, Sung Sik Lee, Kyu-Hyuk Lee, Heesung Do, and JeongGil Ko. Safefac: Video-based smart safety monitoring for preventing industrial work accidents. *Expert Systems with Applications*, 215:119397, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18407–18418. IEEE, 2024.

Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Live: Learning video llm with streaming speech transcription at scale. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29083–29095. IEEE, 2025.

Michele Cournan, Benjamin Fusco-Gessick, and Laura Wright. Improving patient safety through video monitoring. *Rehabilitation Nursing*, 2016.

Junbo Cui, Bokai Xu, Chongyi Wang, Tianyu Yu, Weiyue Sun, Yingjing Xu, Tianran Wang, Zhihui He, Wenshuo Ma, Tianchi Cai, et al. Minicpm-o 4.5: Towards real-time full-duplex omni-modal interaction. *arXiv preprint arXiv:2604.27393*, 2026.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pp. 11198–11201, 2024.

Huang Fang, Mengxi Zhang, Heng Dong, Wei Li, Zixuan Wang, Qifeng Zhang, Xueyun Tian, Yucheng Hu, and Hang Li. Robix: A unified model for robot interaction, reasoning and planning. *arXiv preprint arXiv:2509.01106*, 2025.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118. IEEE, 2025a.

Shenghao Fu, Qize Yang, Yuan-Ming Li, Yi-Xing Peng, Kun-Yu Lin, Xihan Wei, Jian-Fang Hu, Xiaohua Xie, and Wei-Shi Zheng. Vispeak: Visual instruction feedback in streaming videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21778–21788. IEEE, 2025b.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22195–22206, 2024b.

Baoyu Liang, Qile Su, Shoutai Zhu, Yuchen Liang, and Chao Tong. Videvent: A large dataset for understanding dynamic evolution of events in videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 5128–5136, 2025.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 5971–5984, 2024.

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 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 12147–12151. IEEE, 2026.

Kevin Qinghong Lin, Pengchuan Zhang, Joya Chen, Shraman Pramanick, Difei Gao, Alex Jinpeng Wang, Rui Yan, and Mike Zheng Shou. Univtg: Towards unified video-language temporal grounding. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2782–2792. IEEE, 2023.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12585–12602, 2024.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, et al. Ovo-bench: How far is your video-llms from real-world online video understanding? In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18902–18913. IEEE, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24045–24055. IEEE, 2025.

Mengxue Qu, Xiaodong Chen, Wu Liu, Alicia Li, and Yao Zhao. Chatvtg: Video temporal grounding via chat with video dialogue large language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1847–1856. IEEE, 2024.

Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. Timechat: A time-sensitive multimodal large language model for long video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14313–14323. IEEE, 2024.

Bytedance Seed. Seed1. 8 model card: Towards generalized real-world agency. *arXiv preprint arXiv:2603.20633*, 2026.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. Longvu: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*, 2024.

Enxin Song, Wenhao Chai, Tian Ye, Jenq-Neng Hwang, Xi Li, and Gaoang Wang. Moviechat+: Question-aware sparse memory for long video question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Joseph Raj Vishal, Divesh Basina, Aarya Choudhary, and Bharatesh Chakravarthi. Eyes on the road: State-of-the-art video question answering models assessment for traffic monitoring tasks. *arXiv preprint arXiv:2412.01132*, 2024.

Shaohua Wan, Xiaolong Xu, Tian Wang, and Zonghua Gu. An intelligent video analysis method for abnormal event detection in intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4487–4495, 2020.

Haibo Wang, Bo Feng, Zhengfeng Lai, Mingze Xu, Shiyu Li, Weifeng Ge, Afshin Dehghan, Meng Cao, and Ping Huang. Streambridge: Turning your offline video large language model into a proactive streaming assistant. *Advances in Neural Information Processing Systems*, 38:132332–132359, 2026a.

Jiawei Wang, Liping Yuan, Yuchen Zhang, and Haomiao Sun. Tarsier: Recipes for training and evaluating large video description models. *arXiv preprint arXiv:2407.00634*, 2024.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Yueqian Wang, Songxiang Liu, Disong Wang, Nuo Xu, Guanglu Wan, Huishuai Zhang, and Dongyan Zhao. Mmduet2: Enhancing proactive interaction of video mllms with multi-turn reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 23257–23270, 2026b.

Yuxuan Wang, Yueqian Wang, Bo Chen, Tong Wu, Dongyan Zhao, and Zilong Zheng. Omnimmi: A comprehensive multi-modal interaction benchmark in streaming video contexts. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18925–18935. IEEE, 2025b.

Zikang Wang, Boyu Chen, Zhengrong Yue, Yi Wang, Yu Qiao, Limin Wang, and Yali Wang. Videochat-a1: Thinking with long videos by chain-of-shot reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 10467–10475, 2026c.

Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7): 560–576, 2003.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024.

Jiaer Xia, Peixian Chen, Mengdan Zhang, Xing Sun, and Kaiyang Zhou. Streaming video instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31219–31229, 2026.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. Streamingvlm: Real-time understanding for infinite video streams. In *International Conference on Learning Representations*, volume 2026, pp. 61463–61475, 2026.

Antoine Yang, Arsha Nagrani, Paul Hongsuck Seo, Antoine Miech, Jordi Pont-Tuset, Ivan Laptev, Josef Sivic, and Cordelia Schmid. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10714–10726. IEEE, 2023.

Biao Yang, Bin Wen, Boyang Ding, Changyi Liu, Chenglong Chu, Chengru Song, Chongling Rao, Chuan Yi, Da Li, Dunju Zang, et al. Kwai keye-vl 1.5 technical report. *arXiv preprint arXiv:2509.01563*, 2025a.

Haolin Yang, Feilong Tang, Lingxiao Zhao, Xinlin Zhuang, Yifan Lu, Xiang An, Ming Hu, Xiaofeng Zhang, Abdalla Swikir, Junjun He, et al. Streamagent: Towards anticipatory agents for streaming video understanding. *arXiv preprint arXiv:2508.01875*, 2025b.

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, et al. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10807–10816, 2025.

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. In *Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations*, pp. 543–553, 2023.

Ruixuan Zhang, Beichen Wang, Juexiao Zhang, Zilin Bian, Chen Feng, and Kaan Ozbay. When language and vision meet road safety: leveraging multimodal large language models for video-based traffic accident analysis. *Accident Analysis & Prevention*, 219:108077, 2025a.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024.

Yulin Zhang, Cheng Shi, Yang Wang, and Sibei Yang. Eyes wide open: Ego proactive video-llm for streaming video. *arXiv preprint arXiv:2510.14560*, 2025b.

A Data Engine Details

A.1 QA Synthesis Details

We design two distinct pipelines to synthesize streaming QA, motivated by the different temporal structures of the QA types.

The first pipeline focuses on Real-Time and Proactive QA. For each video, we first apply an MLLM to perform scene segmentation and generate scene-level descriptions for each segment, thereby capturing the temporal progression of events throughout the video. Based on the video and its scene-level descriptions, the MLLM then produces candidate QA pairs with associated question and answer timestamps. For Real-Time QA, the question and answer share the same timestamp. For Proactive QA, the question timestamp precedes the answer timestamp. We then ask the MLLM to verify each candidate QA pair using only the video content up to the answer timestamp. For Real-Time QA, the MLLM checks whether the question is reasonable, whether the answer is correctly supported by the video content, and whether the timestamp is accurate. For Proactive QA, the MLLM further checks whether the question can be naturally raised at the question timestamp and whether sufficient information is available by the answer timestamp. Only QA pairs that pass the corresponding verification step are retained.

The second pipeline is designed for Multi-Response QA. For each video, we first apply an MLLM to perform scene segmentation and generate scene-level descriptions together with the corresponding time intervals for each scene. Then, for each video clip spanning a given time interval, we provide its scene description to the MLLM to generate candidate questions and their associated timestamps. Next, each candidate question is checked to determine whether it is reasonable and whether it can produce multiple valid answers at different timestamps within the video clip. Only the questions that satisfy both conditions are retained. For each retained question, we again prompt the MLLM with the corresponding video clip to generate multiple answers and their associated timestamps. Each generated answer is then verified to ensure that it can be correctly inferred at the specified timestamp and that it appropriately answers the question. Finally, the validated answers are retained to construct Multi-Response QA samples.

A.2 QA Refinement Details

We observe that QA samples generated in the QA Synthesis stage still exhibit limited diversity. In particular, Real-Time QA tends to exhibit limited diversity in difficulty levels, while Proactive QA and Multi-Response QA often show limited diversity in question phrasing. To better capture the diversity of difficulty levels and phrasings in real-world user queries, we design tailored refinement strategies for different QA types.

For Real-Time QA, we focus on enriching the diversity in difficulty levels. For each original QA pair associated with a timestamp, we prompt the MLLM to generate four additional questions anchored at the same timestamp as the original question. These questions are designed to span progressively increasing levels of difficulty, from simple perceptual recognition to advanced understanding and reasoning. As a result, we obtain five candidate questions for each timestamp, including the original question and four augmented questions at different difficulty levels. We then sample one question from these five candidates using a balanced sampling ratio. After the question is selected, we prompt the MLLM to generate an answer aligned with the sampled question based on the same video prefix, yielding the final refined QA pair.

For Proactive QA and Multi-Response QA, we focus on increasing the diversity in question phrasing by rewriting each question into different yet semantically equivalent forms. We predefine a set of candidate templates for both Proactive and Multi-Response questions. For each question, we randomly select one template and use an LLM to generate a rewritten version accordingly. During this process, key entities, actions, and temporal references are kept unchanged to ensure consistency with both the original meaning and the video content. In this way, we enhance linguistic diversity without changing the core semantics of the original QA pairs.

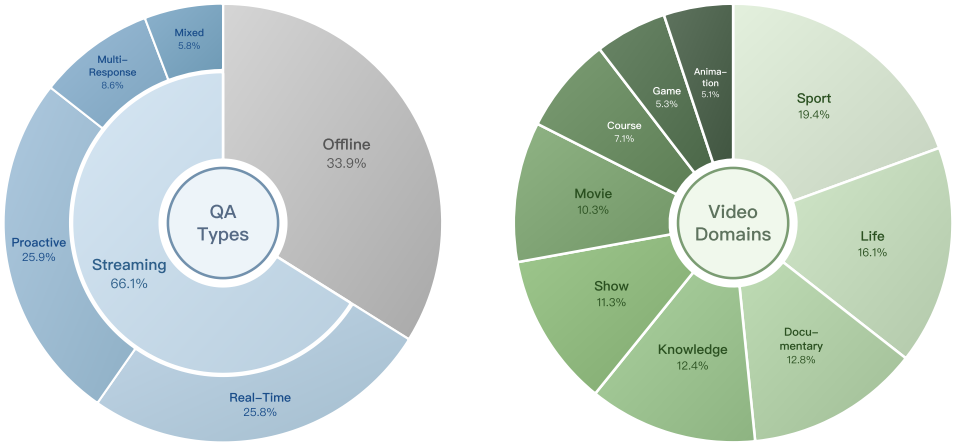


Figure 6: Training data distribution. Left: QA type distribution. Right: Video domain distribution. The training set covers diverse question formats and video domains.

A.3 Quality Verification Details

For Real-Time QA, verification focuses on whether the target answer is supported by the relevant visual evidence available up to the response moment together with the retained QA history. The judge evaluates whether the answer is visually grounded, factually correct, temporally consistent, and free of hallucination. Only samples that satisfy these criteria are retained. For Proactive QA and Multi-Response QA, verification focuses on whether the target answer is temporally appropriate and whether its content is accurate and grounded in the retained video context and QA history. Samples are retained only when both criteria are satisfied.

A.4 Training Data Distribution

The training data distribution is shown in Fig. 6. The left panel shows the distribution of QA types across the training set, while the right panel shows the distribution of video domains. The training set covers diverse question formats and video domains, providing broad supervision for AURA.

B Inference Framework Details

B.1 Context Truncation and KV-Cache Reuse

Following the definition of the Interactive Video Stream Context Management mechanism, context truncation is required once the total context reaches the predefined video window size N. A common approach is to maintain all video chunks in the context as a fixed-length first-in-first-out (FIFO) queue, removing the oldest chunk whenever a new one is added. However, this design causes the context prefix to change continuously, which prevents the reuse of previously computed KV-caches (prefix caching) and significantly reduces inference efficiency. To address this issue, we adopt an improved strategy that allows the video window to slightly extend within a margin of N'. When the window size reaches $N + N'$, we remove the oldest N' video chunks at once, together with the corresponding silent assistant messages. During the insertion of the next N' video chunks, no further chunks are removed, which allows the KV prefix to be reused continuously. Only when the queue length again reaches $N + N'$ do we truncate the window back to N by removing the oldest N' video chunks, and then recompute the KV-cache for the remaining N chunks. For the text QA sliding window, we similarly allow it to extend slightly above M. However, in order to avoid increasing the frequency of KV prefix cache recomputation, we do not set a separate truncation trigger for QA. Instead, whenever the video window is truncated, we simultaneously trim the text QA groups that lie outside the video window to M. Since QA accumulates relatively slowly, the floating range above M will also remain small. Although these two floating-window designs may cause the inference-time context length to

Table 2: Quantitative comparison on OVO-Bench in terms of accuracy (%). We report performance on each fine-grained task, along with the average results for Real-Time Visual Perception (RTVP), Backward Tracing (BT), Forward Active Responding (FAR), and the overall average. The best and second-best results are highlighted in bold and underlined, respectively.

Method	RTVP							BT				FAR				All
	OCR	ACR	ATR	STU	FPD	OJR	Avg	EPM	ASI	HLD	Avg	REC	SSR	CRR	Avg	
Proprietary Models																
GPT-4o	69.8	64.2	71.6	51.1	70.3	59.8	64.5	57.9	75.7	48.7	60.8	73.2	27.6	59.4	53.4	59.5
Gemini-1.5-Pro	85.9	67.0	79.3	58.4	63.4	62.0	69.3	58.6	76.4	52.6	62.5	35.5	74.2	61.7	57.2	63.0
Open-Source Models																
VideoLLM-EyeWO	24.2	27.5	31.9	32.6	44.6	35.9	32.8	39.1	38.5	6.5	28.0	–	–	–	–	–
StreamAgent	71.2	53.2	63.6	53.9	67.3	58.7	61.3	54.8	58.1	25.8	41.7	35.9	48.4	52.0	45.4	49.4
Streamo-7B	77.2	66.1	76.7	45.5	66.3	72.8	67.4	55.6	58.1	33.9	49.2	30.8	57.6	**82.5**	**57.0**	57.9
ViSpeak	75.2	58.7	71.6	51.1	74.3	66.9	66.3	**59.9**	48.7	**64.0**	57.5	33.8	68.5	60.4	54.3	61.1
MiniCPM-o-4.5	73.2	65.1	75.0	59.0	69.3	62.5	67.3	55.2	66.9	45.7	55.9	**43.0**	66.0	58.8	55.9	59.7
Qwen3-VL-8B-Inst.	54.5	58.7	68.5	**70.3**	68.1	49.5	61.6	58.8	41.1	23.1	41.0	21.2	51.3	40.4	37.7	46.8
AURA (Ours)	**89.9**	**79.8**	**80.2**	70.2	**77.2**	**81.5**	**79.8**	54.9	**67.6**	58.6	**60.4**	30.4	**75.8**	61.3	55.8	**65.3**

Table 3: Quantitative comparison on OmniMMI in terms of accuracy (%). We report performance for Dynamic State Grounding (SG), Action Prediction (AP), Multi-turn Dependency Reasoning (MD), Speaker Identification (SI), Proactive Alerting (PA), and the overall average. The best and second-best results are highlighted in bold and underlined, respectively.

Method	SG				AP	MD				SI	PA	All
	1st	2nd	3rd	Avg		1st	2nd	3rd	Avg			
Proprietary Models												
GPT-4o	48.7	17.0	5.6	15.0	39.5	34.3	15.6	7.7	12.3	17.0	✗	16.8
Gemini-1.5-Pro	52.3	19.7	9.4	16.3	43.0	35.0	16.3	7.1	12.0	38.5	✗	22.0
Open-Source Models												
M4	35.7	6.4	1.9	5.7	33.5	35.7	6.4	1.9	1.7	9.0	25.5	15.1
MiniCPM-o-4.5	46.3	8.1	0.9	6.3	25.0	36.0	**13.5**	**5.1**	9.3	17.0	✗	11.5
Qwen3-VL-8B-Inst.	51.3	9.5	0.0	7.7	**36.5**	33.7	11.1	3.5	**9.7**	**29.5**	✗	16.7
AURA (Ours)	**52.7**	**26.4**	**15.0**	**24.0**	32.0	**37.3**	12.8	3.6	7.7	26.0	**37.5**	**25.4**

be slightly larger than that used during training, the excess is small. We do not observe noticeable performance degradation in practice, while the computational cost is significantly reduced.

C Additional Experimental Results

C.1 OVO-Bench Results

As reported in Tab. 2, AURA again obtains the highest overall accuracy of 65.3% on OVO-Bench, improving over the best open-source result from ViSpeak by 4.2%. At a finer granularity, AURA is the best open-source model on 2 of the 3 high-level settings, namely Real-Time Visual Perception (RTVP) and Backward Tracing (BT), where it more than doubles the performance of VideoLLM-EyeWO. On Forward Active Responding (FAR), it trails the best-performing model by only 1.2%. Compared with proprietary models, AURA remains competitive and even achieves a higher overall score than Gemini-1.5-Pro by 2.3% (65.3% vs. 63.0%).

C.2 OmniMMI Results

As shown in Tab. 3, AURA achieves the best overall accuracy of 25.4% on OmniMMI, surpassing all open-source and proprietary models and ranking first on 5 of the 9 fine-grained metrics. Non-streaming models, including MiniCPM-o-4.5 in non-duplex mode, do not support Proactive Alerting (PA), so their PA results are left blank.

Table 4: Offline benchmark comparison for studying the effect of streaming-oriented training on conventional offline video understanding. All results are reported in accuracy (%).

Method	LongVideoBench	MVBench	Video-MME
Qwen3-VL-8B-Inst.	61.9	69.0	68.6
AURA (Ours)	58.8	68.1	65.1

C.3 Inference Performance Details

TTFT Experimental Setup. We implement the AURA inference engine on top of vLLM (Kwon et al., 2023) and evaluate it in a latency-sensitive online video understanding setting. We use the same high-performance GPUs described in Sec. 6.1 to deploy the system. We compare three configurations to isolate the contribution of our context management and cache reuse mechanisms: *w/o sliding window*, which disables the sliding-window context management in Sec. 3.1, *w/o prefix caching*, which disables prefix KV reuse by setting $N' = 1$, and *AURA*, which enables the sliding window and uses $N' = 15$. For all settings, we use batch size 1, stream video continuously for 5 minutes at 2 FPS, and report time-to-first-token (TTFT), defined as the server-side latency between issuing a user query and receiving the first generated text token.

End-to-end Latency Analysis. We conduct independent latency analysis for each system component. For inference deployment, we use two GPUs: one hosts both the ASR service (Qwen3-ASR-1.7B with vLLM as the inference backend) and the TTS service (Qwen3-TTS-12Hz-1.7B-Base with streaming synthesis), while the other hosts the main model (AURA with vLLM serving), with both sliding-window and prefix caching enabled. For ASR, the test input consists of a 9.41-second Chinese speech command in MP3 format. For AURA, the test input consists of a 5-minute video at 2 FPS. For TTS, the test input consists of a 40-character Chinese sentence. ASR and TTS are each evaluated independently over 10 runs on a fully warmed-up engine, whereas TTFT is measured during a 5-minute continuous streaming session to reflect steady-state performance under accumulated multi-turn context.

The ASR component transcribes the 9.41-second audio into text with an end-to-end latency of 84.2 ms. For the main model, the server-side TTFT averages 75.0 ms (p50 = 74.6 ms, p90 = 87.8 ms) over the 5-minute sustained streaming session, where sliding-window context management keeps the active context bounded and prefix caching avoids redundant recomputation across successive queries. During decoding, the generation speed is about 7.3 ms per token (about 137 tokens/s), and each response contains 12.6 generated tokens on average, corresponding to a total decoding time of about 100 ms per response. However, because TTS operates in a sentence-level streaming fashion, audio synthesis can begin once the first sentence has been fully decoded. A typical first sentence is estimated to contain approximately 8 tokens, yielding an estimated first-sentence decode time of about 60 ms. The remaining tokens are decoded in parallel with TTS synthesis and therefore do not contribute to the initial user-perceived latency. For streaming TTS, the first-chunk latency is highly stable at 93.0 ms on average, with a real-time factor (RTF) of 0.42.

C.4 Research Questions

RQ1: Accuracy on Offline Video Benchmarks. To study how streaming-oriented training affects conventional offline video understanding, we compare AURA with its initialization model, Qwen3-VL-8B-Instruct, on three representative offline benchmarks, namely LongVideoBench (Wu et al., 2024), MVBench (Li et al., 2024b), and Video-MME (Fu et al., 2025a). We evaluate all three benchmarks with VLMEvalKit (Duan et al., 2024), adhering to the original benchmark settings except that we use uniform 2 FPS sampling. This experiment is designed to answer a key question in our work: after optimizing the model for continuous streaming observation, selective silence, and delayed response in streaming scenarios, can it still preserve competitive offline understanding ability?

As shown in Tab. 4, AURA achieves 58.8% on LongVideoBench, 68.1% on MVBench, and 65.1% on Video-MME. Compared with its base model (Qwen3-VL-8B-Instruct (Bai et al., 2025)), AURA remains particularly close on MVBench, while showing modest performance drops on LongVideoBench and Video-MME. Overall, these results indicate that streaming-oriented training enhances online interaction ability while largely preserving the model's offline video understanding capability.

Table 5: Ablation study on the training objective over OmniMMI in terms of accuracy (%). We report the averaged performance for Dynamic State Grounding (SG), Action Prediction (AP), Multi-turn Dependency Reasoning (MD), Speaker Identification (SI), Proactive Alerting (PA), and the overall average. The best and second-best results are highlighted in bold and underlined, respectively.

Objective	SG	AP	MD	SI	PA	All
Default Cross-Entropy Loss	<u>22.7</u>	<u>26.0</u>	**7.7**	<u>25.5</u>	<u>0.0</u>	<u>16.4</u>
Silent-Speech Balanced Loss (Ours)	**24.0**	**32.0**	**7.7**	**26.0**	**37.5**	**25.4**

RQ2: Effect of the Training Objective. To evaluate the training objective in Sec. 5.1, we conduct an ablation study on OmniMMI. We train a variant of AURA with the same data, initialization, and training settings, but replace our objective with the default cross-entropy loss that uniformly optimizes all assistant messages. As shown in Tab. 5, replacing our objective with the default loss substantially hurts overall performance. The overall average drops from 25.4% to 16.4%, and PA falls from 37.5% to 0.0%. Clear drops are also observed on State Grounding (SG), Action Prediction (AP), and Speaker Identification (SI).

Closer inspection shows that the model trained with the default loss tends to over-generate `<|silent|>`, remaining silent at every time step in PA. For non-proactive interaction metrics, the degradation is also consistent with our supervision-selection design: under sliding-window truncation, earlier non-silent assistant responses may no longer be fully supported by the preserved visual and history context, so uniformly supervising all assistant messages can introduce insufficiently grounded targets and increase hallucination risk. These results match the goals of our objective. It reweights silent messages to prevent excessive silence and supervises only the final non-silent assistant message to reduce hallucinations.