FORESIGHT: PLANNING FUTURE PERCEPTION IN STREAMING VLMS WITHOUT RETRAINING

Anonymous authors
Paper under double-blind review

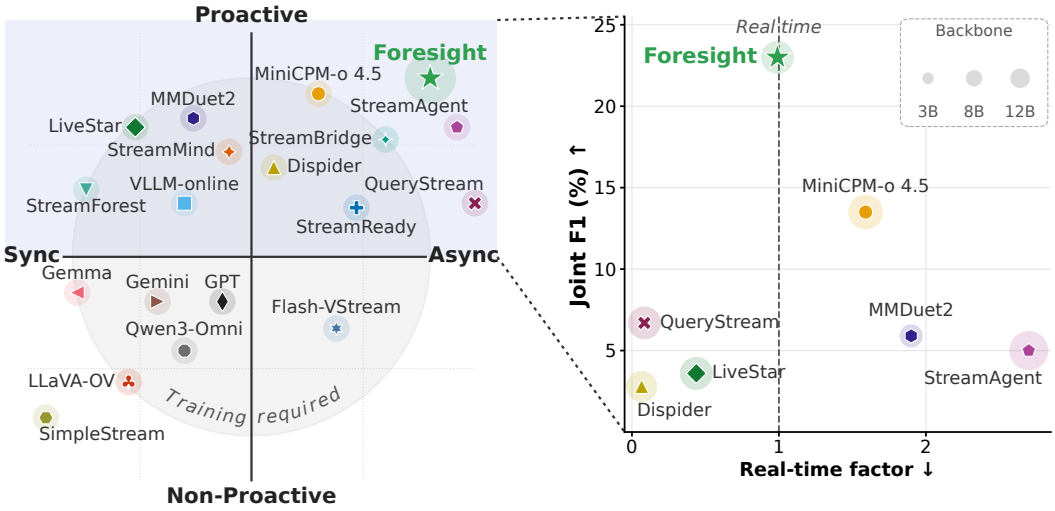


Figure 1: **Foresight shifts streaming VLMs from reactive to proactive inference**. Left: existing methods often require additional training, where the proactive and asynchronous nature are vital for human-AI interactivity. Right: Foresight performs better then training based methods while being proactive, asynchronous, and superior to real time inference on the challanging OmniPro benchmark.

ABSTRACT

Existing streaming vision-language models (VLMs) continuously perceive and reason over visual streams, but their computational pathways remain fixed throughout inference. Consequently, they cannot adapt computation to evolving scene dynamics, where different future events demand different levels and forms of perception. We show that streaming VLMs inherently possess the ability to anticipate the immediate future, and leverage this capability to dynamically configure future computation in a training-free manner. Realizing such anticipatory computation, however, is very challenging: future anticipation must be sufficiently reliable to guide computation, planning must run concurrently with streaming inference, and online reconfiguration must incur negligible overhead.

To address these challenges, we introduce FORESIGHT, a dual-stream architecture comprising two Siamese LLMs with shared weights, input encoders, and KV cache. The first LLM continuously processes incoming tokens, while the second runs ahead of the stream to anticipate future context, plan future computation, and generate task responses without interrupting streaming inference. Each plan decides when to reason next, what to check then, and how densely to sample, keeping transient evidence separate from persistent control. The resulting computation plan is executed online through an efficient reconfiguration protocol with schema-guided decoding and lightweight diff-based updates, enabling dynamic adaptation with low overhead. With a frozen Qwen3-VL-8B backbone, FORESIGHT achieves 23.0 mean joint F1 on OmniPro Online evaluation beating strongest trained baseline by 9.5%, while improving the backbone by 6.7 on StreamingBench and 15.4 on OVO-Bench, with the largest gain of 18.7 when evidence arrives later in the video stream. Our source code will be made publicly available.

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.

1 INTRODUCTION

Vision-language models (VLMs) (Lin et al., 2024a; Li et al., 2025a; Wang et al., 2024b) have made substantial progress in understanding complex visual content, but most existing models operate offline: they receive a fixed input, perform a predetermined computation, and produce an answer. In contrast, real-world visual intelligence is inherently streaming. An agent observing a live video must continuously process new observations while deciding *when* the available evidence is sufficient, *what* information should be examined next, and *how much* computation should be allocated to the evolving scene. These decisions are especially important in human-AI interaction, where the system must respond to events *proactively*, without being explicitly prompted, and *asynchronously*, staying engaged with the user and the stream while deeper computation runs in the background.

Existing streaming VLMs meet these requirements only in part. Many learn when to respond (Chen et al., 2024a; Wang et al., 2025b; 2026; Qian et al., 2025; Azad et al., 2026), reduce the cost of each step through frame sampling or token pruning (Chen et al., 2024b; Yao et al., 2025), or use memory compression (Qian et al., 2024; Xu et al., 2025; Chen et al., 2026); some also separate lightweight stream monitoring from heavier asynchronous reasoning (Qian et al., 2025; Wang et al., 2025a; Ding et al., 2025; Li et al., 2025b; Yakushev et al., 2025). Figure 1 sorts these methods according to whether they are proactive and whether they are asynchronous. Yet even methods that are both proactive and asynchronous still control computation reactively, based on what has already arrived rather than on what is expected. In an evolving scene, the best next step is often to wait for the right evidence. Existing systems leave that wait to either a fixed schedule (Yang et al., 2025a) or a learned trigger (Wang et al., 2025a; Ding et al., 2025; Azad et al., 2026).

We argue that this wait should be planned by the model itself. In fact, a streaming VLM already understands the scene well enough to know what it is still missing, so it can also estimate when that information will appear and what to check once it does. We therefore treat anticipation not as a prediction target but as a way to control computation: at each reasoning step, the model identifies what remains uncertain, decides when to look again, and sets how densely to sample until then. Streaming inference then becomes a closed loop rather than a fixed sequence of computations, with the model's current state deciding how it processes the future. Figure 2 sketches this loop: one copy of a frozen VLM keeps ingesting the stream, while a second copy reads the same state in the background and writes the plan. Because planning never pauses perception and requires no retraining, this places our approach in the proactive, asynchronous, training-free corner of Figure 1.

Turning anticipation into control raises three planning challenges: (i) reliability: wrong anticipation misguides reasoning; (ii) concurrency: planning must run alongside perception; and (iii) efficiency: its cost must not negate adaptive inference gains. We address these with **FORESIGHT**, a training-free perception-planning architecture built on a frozen VLM. For reliability, each plan separates a *transient* evidence state, rebuilt from current observations at every step, from a *persistent* control update that records when to check next and which question to revisit. The responses are further gated by hysteresis on the model's own evidence confidence, so small fluctuations do not trigger premature or repeated outputs. For concurrency, we use a Siamese architecture comprising Ingest and Think copies. The Ingest copy is the only writer to the shared KV cache, while the Think copy plans from a read-only snapshot, so planning never blocks ingestion. For efficiency, the runtime fills in the plan's fixed structure itself, the model decodes only the values that require reasoning, and a live executor applies the result to the input gate, the encoder, and the cache.

We evaluate FORESIGHT with a frozen Qwen3-VL-8B backbone. On OmniPro Online, FORESIGHT reaches 23.0 mean F1, against 13.5 for the strongest baseline, MiniCPM-o 4.5. Figure 1 (right) shows that this accuracy does not come at the cost of speed. FORESIGHT also helps general streaming understanding, improving its backbone by 6.7 points on StreamingBench, with SOTA results, and by 15.4 points on OVO-Bench. The largest gain, 18.7 points, is on Forward Active Responding, where the answer appears only later in the video. This is the setting our argument targets: the right move is to hold the question and check again once the evidence arrives. Our contributions are:

- We formulate **anticipatory computation** for streaming VLMs, in which the model uses its current understanding of the scene to decide when to reason next, what to check then, and how densely to sample until then.

- We introduce **FORESIGHT**, a training-free architecture in which one frozen VLM perceives and plans over a shared causal state, so that planning runs asynchronously without pausing perception, and plans are decoded and executed with little overhead.
- We show that this inference-time controller leads all compared systems on proactive responding, improves streaming understanding, and helps most where evidence arrives later, supporting anticipation as a way to control computation, not just prediction.

2 RELATED WORK

Streaming and proactive vision-language models. Early streaming VLMs interleaved video frames with text and learned when to stay silent and when to speak (Chen et al., 2024a; Wang et al., 2025b). Subsequent systems strengthened this streaming interaction loop in complementary ways. MMDuet2 (Wang et al., 2026) improves the response-timing decision through supervised fine-tuning followed by reinforcement learning, whereas StreamChat (Liu et al., 2024) refreshes the visual context at every decoding step to keep generation grounded in the evolving stream. Later systems became proactive in egocentric and live video (Zhang et al., 2025b; Yu et al., 2025; Yang et al., 2025c; 2026; Yan et al., 2026), always-on systems were built around them (Lu et al., 2026; Yao et al., 2026), and MiniCPM-o 4.5 (Cui et al., 2026) now handles vision, audio, and speech in full duplex. To find whether model has seen enough to answer, LiveStar (Yang et al., 2025c) decodes again when new frames stop matching its current description. Dispider (Qian et al., 2025) trains a lightweight classifier to give the decision whether to respond or not. StreamReady (Azad et al., 2026) goes a step further and learns a readiness token that is penalized for answering too early or too late. QueryStream (Zhang et al., 2026) uses the similarity score of query and videostream to make triggering decision without training.

Asynchronous Computation. When the model is giving response to the user, it should be also asynchronously watching the video stream for real time computer-human interactions. Further running a large VLM on every frame to find if enough information to answer a query has been seen is expensive, so several systems separate cheap proactivity decision from costly response generation. In Dispider (Qian et al., 2025), a lightweight trained classifier watches the stream and decides when to respond while a larger model answers asynchronously. StreamBridge (Wang et al., 2025a) and StreamMind (Ding et al., 2025) call the language model only when a small trained gate fires. LION-FS (Li et al., 2025b) pairs a fast path that decides frame by frame whether to respond with a slow path that adds detail once it does. Reasoning has received similar attention. VST (Guan et al., 2026) spreads reasoning across playback, R3-Streaming (Liu et al., 2026b) learns when to route a query to a stronger model, and Asynchronous Reasoning (Yakushev et al., 2025) lets a model keep reading input while it thinks, without extra training. Like adaptive computation and dynamic networks more broadly (Graves, 2016; Huang et al., 2018; Han et al., 2022; Raposo et al., 2024), these systems react to the present and decide how much compute should it perform on input that has just arrived. Unlike other asynchronous systems that use different LLMs or encoders which have to duplicate the task of video encoding and processing, FORESIGHT uses asynchronous twin VLMs with shared KV-cache.

Anticipatory Computation Using anticipation to steer computation is much less explored. Action anticipation has its own models and benchmarks (Girdhar & Grauman, 2021; Grauman et al., 2022; Zhao et al., 2024), but there the prediction is the final output. To our knowledge, StreamAgent (Yang et al., 2025a) is the only proactive streaming method that feeds its predictions back into computation. It predicts when and where task-relevant events will appear so that perception can be pointed at them. Its planning, however, runs on a fixed schedule: every clip triggers a caption, a set of candidate plans to be scored by unreliable LLM judge, and a trigger check. In FORESIGHT, one plan is generated by the VLM that decides when the next reasoning step happens, carries the question that step should resolve, and sets how densely to sample in between.

Pruning. Another way to lower per-step cost is to spend fewer tokens on each frame. FastV (Chen et al., 2024b) drops visual tokens that get little attention in the early LLM layers, and DyCoke and PruneVid (Tao et al., 2025; Huang et al., 2025) merge tokens repeated across frames. Streaming systems do the same online, keeping what changed (Yao et al., 2025) or what matches the query (Zhang et al., 2026). CoPE-VideoLM (Sarkar et al., 2026) skips most of the encoding instead: its delta encoder turns the codec's motion vectors and residuals into eight tokens per P-frame, so only keyframes go through the full image encoder. All of these trim frames that have already arrived. VisionZip (Yang et al., 2025b) uses video context only to prune tokens while QueryStream (Zhang et al., 2026) also utilizes query for token pruning. StreamAgent (Yang et al., 2025a) proposes giving

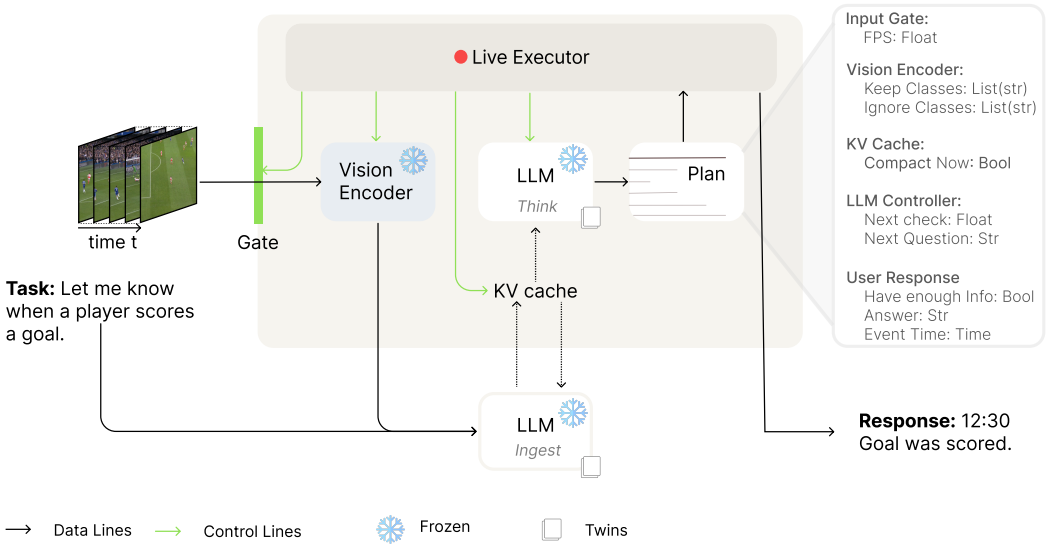


Figure 2: **FORESIGHT overview.** One frozen VLM runs as two twins over a shared KV cache C_t. The *Ingest LLM* (perception) appends every frame admitted by the Vision Gate to C_t and never stops. When a check is due, the *Think LLM* (planning) reads a snapshot of C_t and writes a plan $\pi_t = (s_t, u_t)$ (right panel): an evidence state s_t and a control update u_t. The *Live Executor* applies u_t along the green control lines and emits the answer once the evidence is sufficient.

model agentic tool calling for pruning. FORESIGHT decides beforehand how densely to sample the frames still to come and then uses the model's recommendation of which tokens should be pruned which is much more informed about the task and activities happening in the video.

Self-programming and Reconfiguration. A separate line of work lets a model write the procedure that runs it. In visual programming, a language model turns each query into a short program that calls vision modules (Gupta & Kembhavi, 2023; Surís et al., 2023). Agents interleave reasoning with tool calls whose results change what the model sees next (Yao et al., 2023; Schick et al., 2023), and VideoAgent (Wang et al., 2024c) uses this loop to pick which frames of a long video to inspect. Other systems improve from feedback, refining their own outputs, prompts, or pipelines (Shinn et al., 2023; Madaan et al., 2023; Wang et al., 2024a; Khattab et al., 2024), and some write and rewrite agent code (Zelikman et al., 2024; Hu et al., 2025; Yin et al., 2025). All of them reconfigure over input that stays fixed while they think. FORESIGHT applies a lightweight form of the same idea to a video stream where the rate of change of activities dictates how it should be processed.

3 FORESIGHT

Foresight is a training-free method that equips an off-the-shelf VLM for proactive streaming tasks. It combines asynchronous perception, planning, and response generation, dynamically reconfiguring computation according to the evolving activity in the stream to support real-time human–computer interaction. Here, Section 3.1 explains the architecture illustrated in Figure 2, while Section 3.2 describes the system's dynamic behavior illustrated in Figure 3 and Algorithm 1.

3.1 ARCHITECTURAL OVERVIEW

Live Executor. As shown in Figure 2 the Live Executor takes the plan and reconfigures the system via python program. It maintains the memory of what event happened at what time.

Vision Gate. It samples the stream at the frame rate set by the plan: sparsely while nothing relevant is happening, and more densely when the planner expects an event, so that the moments that matter are captured. Frames that are not admitted are never encoded, which saves computation during quiet stretches of the video and intervals irrelevant to the task at hand.



Figure 3: **FORESIGHT on one stream.** Task: "Let me know when a player scores a goal." Time runs left to right, and the red line marks the goal. The *Ingest* never pauses (Alg. 1, lines 3–6), and the cache grows slowly at 1 fps and faster at 8 fps. The *Think* LLM wakes only at scheduled checks (green bars; line 8), and the interval Δ is set by the previous plan (line 17). Early plans track only the player and ball. As a shot develops, a plan raises sampling to 8 fps and adds the goalkeeper and net; later plans compact the cache (thus KV-cache drops). Just after the goal, the evidence is sufficient and the answer is emitted; the plan then drops the goalkeeper and net, and sampling returns to 1 fps.

Vision Encoder. The Vision Encoder is the VLM's own frozen image encoder that turns each admitted frame into visual tokens and then takes `keep class` and `ignore class` to dynamically prune tokens of irrelevant parts of the video as configured by the Live Executor via the generated plan.

Ingest LLM. The Ingest LLM writes the task query and then the visual tokens of each admitted frame into the shared KV cache C_t, in the order they arrive. It runs independently of the Think LLM and never waits for a plan, so perception continues while planning happens. It is also the only component that writes to C_t, which keeps the cache in one consistent temporal order.

Shared KV-cache. It acts as the communication channel between the continuously running Ingest LLM and Think LLM so that the Think LLM can simply take the context accumulated so far and quickly generate a plan instead of encoding the inputs. To stay within a fixed memory budget, old entries are evicted with a sliding window, or compacted when the plan asks for it.

Think LLM. The Think LLM reads the shared KV cache C_t and writes a plan π_t that reconfigures the system for the stream ahead. The Live Executor wakes the Think LLM at the time the last plan chose. It also passes along the question that plan left open, so the Think LLM knows what to look for. The Think LLM reads a copy of C_t and never writes to it. This keeps its own reasoning out of the visual history. Instead, the Live Executor keeps a record of what it has already observed and reported (`seen` and R_t) and passes it to later plans.

Plan. Each time the Think LLM runs, it writes a plan

$$\pi_t = \big(s_t, u_t\big). \tag{1}$$

Control Fields (u_t). `FPS` field configures Vision Gate to sample frames at adaptive frame rate while `keep class` and `ignore class` configure which objects in the stream should get attention for performing the task. `Compact Now` sends signal to KV-cache to compact the context.`Next Check` and `Next Question` configure Think LLM by controlling when it is invoked next for updating plan and what it should pay attention to then respectively.

User Response (s_t). `Have enough info` says whether the evidence is sufficient to proactively give response. If it is, `Event time` gives when the event happened and `Answer` to the user.

3.2 DYNAMIC BEHAVIOUR

Figure 3 traces FORESIGHT on one stream. The Vision Gate admits frames at the current frame rate, and the Vision Encoder turns them into visual tokens, dropping the tokens of object classes the plan ignores. The Ingest LLM appends the task query and these tokens to the shared KV cache C_t without pausing. Meanwhile, at each scheduled check, the Think LLM reads a snapshot of C_t and writes a plan π_t. The Live Executor then applies the plan. It compacts the cache if `Compact Now`

Algorithm 1 FORESIGHT: anticipatory control for streaming VLM inference

Require: frozen VLM $\mathcal{M}$, visual stream V, task instruction I
1: $C_0 \leftarrow \text{INITIALIZECONTEXT}(\mathcal{M}, I)$
2: $U \leftarrow U_0$; $R_0 \leftarrow \varnothing$; $\tau_{\text{next}} \leftarrow \Delta_0$

The following two roles run concurrently:
Streaming perception — sole writer of C_t
3: **for** each admitted observation (t, x_t) **do**
4: $C_t \leftarrow \text{APPEND}(C_{t^-}, \mathcal{M}_{\text{enc}}(x_t))$
5: $\text{PUBLISH}(t)$
6: **end for**

Anticipatory planning — snapshot reader
7: **while** stream is active **do**
8: **wait until** published video time $t \geq \tau_{\text{next}}$
9: $C_t^{\text{snap}} \leftarrow \text{SNAPSHOT}(C_t)$
10: $\pi_t = (s_t, u_t) \leftarrow \text{PLAN}(\mathcal{M}, C_t^{\text{snap}}, U.q, R_t)$ ▷ Alg. S1
11: $U \leftarrow \text{MERGE}(U, u_t)$
12: $\text{APPLYSAMPLINGCONTROL}(U.\text{fps})$
13: **if** $\text{SHOULDEMIT}(s_t, p_{\text{hit},t}, t)$ **then** ▷ Alg. S2
14: $\text{EMIT}(s_t.\text{answer})$
15: $R_t \leftarrow \text{UPDATEHISTORY}(R_t, t, s_t.\text{answer})$
16: **end if**
17: $\tau_{\text{next}} \leftarrow t + U.\Delta$
18: **end while**

is set, and it passes any new `FPS`, `Keep Classes`, or `Ignore Classes` to the Vision Gate and Vision Encoder, which adjust frame- and token-level pruning for the frames to come.

Algorithm 1 gives the same procedure for a frozen VLM $\mathcal{M}$, a visual stream V, and a task instruction I. Two loops run concurrently. The perception loop (lines 3–6) encodes each admitted frame x_t with $\mathcal{M}_{\text{enc}}$ and appends it to the causal state, turning C_{t^-} into C_t. The planning loop (lines 7–18) waits until video time reaches the scheduled check τ_{next}, takes a read-only snapshot C_t^{snap}, and plans from it together with the open question $U.q$ and the response history R_t. It merges the control update into U and applies it. If the output gate lets the answer through, the answer is sent to the user and added to R_t. Finally, the next check is set to $t + U.\Delta$. The controller starts from default U_0 and Δ_0.

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

Model. FORESIGHT wraps a single off-the-shelf VLM, Qwen3-VL-8B (Bai et al., 2025a), whose weights stay frozen for every experiment.

Rather than feeding individual frames, we group consecutive frames into short clips of two frames each before passing them to the model. This lets us use Qwen3-VL's native video processing operations directly, instead of treating each frame as an isolated image. The system instruction and task specification are prefilled once at the start, and each new frame is added with a video timestamp.

Control fields persist in a standing controller state U, updated by

$$U \leftarrow \text{merge}(U, u_t). \tag{2}$$

The merge is partial: a field the plan leaves out keeps its value in U, so a plan can move the next check without resetting the frame rate or the open question. How the plan is decoded and the thresholds are fitted is described in Section 4.1.

Plan Decoding. Configuration fields of the plan, such as when to check next and which question to revisit, are decoded autoregressively. Boolean decisions are not decoded as text: at the field's position we read the logits of the `yes` and `no` tokens and use their relative ratio, $p = e^{z_{\text{yes}}}/(e^{z_{\text{yes}}} + e^{z_{\text{no}}})$, as a probe that triggers a response when an event of relevance occurs. The first planning step

decodes the full plan; each later step decodes only a diff against the current plan, rewriting the fields that change and keeping the rest, which keeps plan updates light throughout the stream.

Live Executor. The live executor takes each decoded plan and reconfigures the system at that moment. Its execution code receives the plan values as arguments, for example when the planner should wake next, which question it should revisit, and, when adaptive sampling is enabled, how densely the stream is sampled, and applies them to the running pipeline. The configuration is therefore dynamic: it changes as the video progresses and each new plan arrives.

Pruning. The plan can also tell the vision encoder what to look at. Keep Classes lists the object classes that matter for the task and Ignore Classes those that do not, and the encoder drops visual tokens from the ignored classes before they reach the cache. This keeps the cache focused on task-relevant evidence without retraining the encoder.

Proactive Response Triggering and Threshold Fitting. Prompts and decoding settings are fixed within each benchmark (Appendix B.2). The only values fitted to data are the thresholds that decide when to respond. We split each benchmark at random into 20% for calibration and 80% for evaluation, and on the calibration part we fit, per task, the evidence threshold θ_{task}, the gate's firing and re-arming thresholds $\theta_{\text{hi}} > \theta_{\text{lo}}$, and a refractory interval between responses. We also pick per task whether the gate fires every time the evidence is above threshold or only when it first crosses it. To avoid reporting the same event twice, a new response is dropped if the event time the model predicts falls within the refractory interval of an earlier one. All FORESIGHT results are on the held-out 80%; baseline numbers are taken from the full evaluation sets.

Compute. All experiments are run on four NVIDIA GH200 Grace Hopper GPUs.

4.2 BENCHMARKS

We evaluate FORESIGHT on three streaming benchmarks: OmniPro (Zhao et al., 2026) for proactive responding, and StreamingBench (Lin et al., 2024b) and OVO-Bench (Niu et al., 2025) for general streaming video understanding.

OmniPro Online Mode. In the Online mode, the task is given once at the start and the system must decide for itself when to respond. A response counts only if it is correct and arrives within ± 3 s of the annotated event, and we report joint F1 over nine task types under the official protocol (Table 1). Since our backbone takes video only, we use the subset of OmniPro that does not need audio. FORESIGHT reaches a mean timing F1 of 43.7 and a content accuracy of 54.6 on its on-time responses; the per-task split is in Appendix C.1, and Figure 1 (right) plots joint F1 against the real-time factor (compute time per second of video).

Table 1: **OmniPro Online Mode, joint F1 (%)** on the subset that does not need audio. Bold is best, underline second best.

		Perception				Comprehension			Reasoning		
Model	**Params**	Event-Alert	Target-Ground	State-Monitor	Snap.-Count	Cond.-Alert	Cum.-Count	Event-Narr.	Dedup.-Count	Step-Inst.	**Mean**
LiveStar (Yang et al., 2025c)	8B	9.7	0.8	0.0	0.0	14.7	0.0	1.6	0.0	6.0	3.6
MMDuet2 (Wang et al., 2026)	3B	12.5	5.3	<u>14.9</u>	<u>11.2</u>	21.4	5.3	3.7	<u>12.7</u>	<u>14.7</u>	11.3
MiniCPM-o 4.5 (Cui et al., 2026)	9B	**50.1**	<u>12.6</u>	9.8	8.0	<u>23.8</u>	<u>9.0</u>	3.0	3.8	1.4	<u>13.5</u>
Dispider† (Qian et al., 2025)	7B	2.3	0.0	0.0	0.0	8.5	0.3	4.5	0.1	9.7	2.8
StreamAgent (Yang et al., 2025a)	8B	6.8	1.4	4.0	1.9	11.3	0.9	3.7	8.1	7.1	5.0
QueryStream (Zhang et al., 2026)	8B	8.2	1.4	3.1	1.0	17.6	6.6	<u>7.2</u>	7.1	8.3	6.7
FORESIGHT	8B	<u>44.7</u>	**17.9**	**16.4**	**19.1**	**33.8**	**16.3**	**11.5**	**29.2**	**18.5**	**23.0**

StreamingBench StreamingBench tests whether the same controller helps beyond proactive tasks. Its 18 tasks fall into three groups: what is on screen now (Real-Time Visual Understanding, RTVU),

how sound and vision fit together (Omni-Source), and what depends on earlier parts of the stream (Contextual). Table 2 summarizes the results, and per-task results are in Appendix D.1.

Table 2: **StreamingBench, accuracy (%).** Bold is best, underline second best.

Method	FPS	RTVU ↑	Omni-Source ↑	Contextual ↑	Overall ↑
VideoLLM-Online (Chen et al., 2024a)	2	35.99	28.45	26.55	32.48
Flash-VStream (Zhang et al., 2024)	1	23.23	26.00	24.12	24.04
Dispider (Qian et al., 2025)	1	67.63	35.66	33.61	53.12
TimeChat-Online‡ (Yao et al., 2025)	1	75.36	36.96	35.63	58.11
StreamAgent (Yang et al., 2025a)	1	74.28	36.26	34.62	57.02
Qwen3-VL-8B-Instruct* (Bai et al., 2025a)	–	74.10	41.90	39.80	59.30
StreamBridge (Wang et al., 2025a), LLaVA-OV	1	68.39	26.60	31.74	50.96
LLaVA-OV + Stream-IT	1	70.92	26.30	29.21	51.73
StreamBridge (Wang et al., 2025a), Qwen2-VL	1	72.01	31.30	36.57	55.09
Qwen2-VL + Stream-IT	1	77.04	24.10	32.55	55.39
StreamForest‡ (Zeng et al., 2025)	1	77.14	38.05	36.34	60.80
MiniCPM-o 4.5 (Cui et al., 2026)	–	<u>78.20</u>	<u>42.10</u>	**44.50**	<u>62.70</u>
FORESIGHT	adaptive	**79.30**	**48.70**	<u>43.80</u>	**66.00**

OVO-Bench OVO-Bench groups its twelve tasks by where the evidence lies: on screen now (Real-Time), earlier in the video (Backward), or still to come (Forward). Table 3 summarizes the results, and per-task results are in Appendix E.

Table 3: **OVO-Bench, accuracy (%).** Bold is best, underline second best.

Method	Frames	Real-Time ↑	Backward ↑	Forward ↑	Overall ↑
Human	native	93.20	92.30	92.90	92.80
Flash-VStream (Zhang et al., 2024)	1 fps	28.37	27.38	45.09	33.61
Dispider (Qian et al., 2025)	1 fps	54.55	36.06	34.72	41.78
TimeChat-Online (Yao et al., 2025)	1 fps	58.60	42.00	36.40	45.60
StreamForest (Zeng et al., 2025)	1 fps	61.20	52.02	53.49	55.57
QueryStream (Zhang et al., 2026)	–	61.40	42.10	39.03	47.51
StreamAgent (Yang et al., 2025a)	1 fps	61.30	41.70	45.40	49.40
Qwen3-VL-8B-Instruct* (Bai et al., 2025a)	–	61.60	41.00	37.70	46.80
MiniCPM-o 4.5 (Cui et al., 2026)	1 fps	67.30	<u>55.90</u>	<u>55.90</u>	59.70
StreamBridge (Wang et al., 2025a), Qwen2-VL-7B + Stream-IT	1 fps	<u>71.30</u>	**68.05**	48.36	**62.57**
FORESIGHT	adaptive	**75.02**	55.10	**56.37**	<u>62.16</u>

Ablations. Table 4 removes one plan field at a time on OmniPro Online, with everything else unchanged. The plan loop itself is essential: if the executor ignores the plan, or the plan keeps only the answer, the model almost never responds and joint F1 drops from 23.0 to below 1. Removing any single field changes joint F1 by less than three points, but this hides a trade-off between tasks: without the scheduling fields, instant alerts improve while tasks that follow an activity over time, such as counting and state monitoring, get worse.

Table 4: **Plan-field ablation on OmniPro Online (%).** Components and field names follow the plan in Figure 2; each row removes the named field from FORESIGHT and keeps everything else fixed.

Component	Plan field removed	Time F1	Joint F1
Full FORESIGHT (as reported)		43.7	23.0
Think LLM	Next Check	43.0	18.9
Think LLM	Next Question	43.3	19.4
Think LLM	Next Check + Next Question	43.2	18.6
Vision Gate	FPS	43.5	23.0
Vision Encoder	Keep Classes	42.7	22.7
Vision Encoder	Ignore Classes	43.5	22.9
KV Cache	Compact Now	44.3	23.4
Live Executor	all fields (plan ignored)	4.7	0.7
User Response	all but Answer	4.0	0.3

5 CONCLUSION

We presented FORESIGHT, a training-free framework that turns a frozen, off-the-shelf VLM into a proactive and asynchronous streaming system. Our central argument is that a streaming VLM already understands a scene well enough to know what it is still missing, and can therefore plan its own future perception instead of reacting only to what has already arrived. FORESIGHT realizes this idea with two copies of the same model over a shared KV cache: one keeps ingesting the stream, while the other plans in the background when to reason next, what to check then, and how densely to sample, so that planning never pauses perception. The experiments support this design. With a frozen Qwen3-VL-8B backbone, FORESIGHT reaches 23.0 joint F1 on OmniPro Online, against 13.5 for the strongest trained baseline, while running close to real time. It also achieves the best overall score on StreamingBench and improves its backbone by 15.4 points on OVO-Bench, with the largest gain, 18.7 points, on Forward Active Responding, i.e. the setting where the answer appears only later in the stream. Because no weights are updated, the same controller can directly benefit from stronger backbones as they become available. More broadly, our results suggest that anticipation is most valuable not as a prediction to be scored, but as a way for a model to decide how to spend its own computation on what comes next.

6 LIMITATIONS AND FUTURE WORK

The accuracy of FORESIGHT's anticipation and temporal event localization depends on the capacity of the underlying VLM. Because the controller relies heavily on these abilities to decide when to reason and respond, reliable operation requires a VLM with strong future anticipation and temporal grounding. Future work should improve these capabilities through stronger backbones or dedicated inference-time mechanisms. After an initial false trigger, the model can occasionally continue reporting an event even when subsequent observations do not support it. Providing the Think LLM with a history of what has been observed and already reported reduces this behavior substantially, but does not eliminate it. The controller also remains sensitive to prompt wording, motivating future work on more prompt-robust control policies.

7 AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing parts of the evaluation and inference code and in correcting LaTeX formatting. We did not use generative AI tools to develop the conceptual framework, propose hypotheses, design the methodology or experiments, or interpret results; generating synthetic data, mathematical claims, proofs, and translation are not applicable to this work. Separately, as part of the benchmark protocol, GPT-4o-mini is used as an automatic judge of content correctness for Event-Narration and Step-Instruction (Appendix C.1). All AI-assisted code was reviewed and tested by the authors, and we take full responsibility for the content of this work.

REFERENCES

Anthropic. Introducing Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, June 2024.

Shehreen Azad, Vibhav Vineet, and Yogesh S. Rawat. Streamready: Learning what to answer and when in long streaming videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 40494–40504, 2026.

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*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

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

Joya Chen, Zeyun Zhong, and Mike Zheng Shou. StreamTTT: Reconciling real-time perception and long-term memory in streaming VLMs. *arXiv preprint arXiv:2608.13416*, 2026.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision (ECCV)*, 2024b.

Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to GPT-4V? closing the gap to commercial multimodal models with open-source suites. *Science China Information Sciences*, 67 (12):220101, 2024c.

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

Junbo Cui, Bokai Xu, Chongyi Wang, Tianyu Yu, Weiyue Sun, Yingjing Xu, Tianran Wang, Zhihui He, Wenshuo Ma, Tianchi Cai, Jiancheng Gui, Luoyuan Zhang, Xian Sun, Fuwei Huang, Moye Chen, Zhuo Lin, Hanyu Liu, Qingxin Gui, Qingzhe Han, Yuyang Wen, Huiping Liu, Rongkang Wang, Yaqi Zhang, Hongliang Wei, Chi Chen, You Li, Kechen Fang, Jie Zhou, Yuxuan Li, Guoyang Zeng, Chaojun Xiao, Yankai Lin, Xu Han, Maosong Sun, Zhiyuan Liu, and Yuan Yao. MiniCPM-o 4.5: Towards real-time full-duplex omni-modal interaction, 2026. URL https://arxiv.org/abs/2604.27393.

Xin Ding, Hao Wu, Yifan Yang, Shiqi Jiang, Qianxi Zhang, Donglin Bai, Zhibo Chen, and Ting Cao. Streammind: Unlocking full frame rate streaming video dialogue through event-gated cognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13448–13459, 2025.

Jiajun Fei, Dian Li, Zhidong Deng, Zekun Wang, Gang Liu, and Hui Wang. Video-CCAM: Enhancing video-language understanding with causal cross-attention masks for short and long videos. *arXiv preprint arXiv:2408.14023*, 2024.

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.

Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18995–19012, 2022.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Yiran Guan, Liang Yin, Dingkang Liang, Jianzhong Ju, Zhenbo Luo, Jian Luan, Yuliang Liu, and Xiang Bai. Video streaming thinking: Videollms can watch and think simultaneously. In *European Conference on Computer Vision (ECCV)*, 2026.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 44(11):7436–7456, 2022.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2018.

Xiaohu Huang, Hao Zhou, and Kai Han. PruneVid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *Transactions on Machine Learning Research (TMLR)*, 2025a.

Wei Li, Bing Hu, Rui Shao, Leyang Shen, and Liqiang Nie. LION-FS: Fast & slow video-language thinker as online video assistant. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3240–3251, 2025b.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. VILA: On pre-training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26689–26699, 2024a.

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

Jiajun Liu, Yibing Wang, Hanghang Ma, Xiaoping Wu, Xiaoqi Ma, Xiaoming Wei, Jianbin Jiao, Enhua Wu, and Jie Hu. Kangaroo: A powerful video-language model supporting long-context video input. *International Journal of Computer Vision (IJCV)*, 134(3):114, 2026a.

Jihao Liu, Zhiding Yu, Shiyi Lan, Shihao Wang, Rongyao Fang, Jan Kautz, Hongsheng Li, and Jose M. Alvarez. StreamChat: Chatting with streaming video, 2024.

Jinming Liu, Jianguo Huang, Zhaoyang Jia, Jiahao Li, Xiaoyi Zhang, Zongyu Guo, Bin Li, Wenjun Zeng, Yan Lu, and Xin Jin. An efficient streaming video understanding framework with agentic control, 2026b. R3-Streaming.

Xudong Lu, Yang Bo, Jinpeng Chen, Shuhan Li, Xintong Guo, Huankang Guan, Fang Liu, Dunyuan Xu, Peiwen Sun, Heyang Sun, Rui Liu, and Hongsheng Li. AURA: Always-on understanding and real-time assistance via video streams. *arXiv preprint arXiv:2604.04184*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. OVO-Bench: How far is your video-llms from real-world online video understanding? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18902–18913, 2025. doi: 10.1109/CVPR52734.2025.01761.

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

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24045–24055, 2025. doi: 10.1109/CVPR52734.2025.02239.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Sayan Deb Sarkar, Rémi Pautrat, Ondrej Miksik, Marc Pollefeys, Iro Armeni, Mahdi Rad, and Mihai Dusmanu. CoPE-VideoLM: Leveraging codec primitives for efficient video language modeling, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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. In *International Conference on Machine Learning (ICML)*, pp. 54582–54599, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. DyCoke: Dynamic compression of tokens for fast video large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research (TMLR)*, 2024a.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-4406.

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*, 2024b.

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

Yueqian Wang, Xiaojun Meng, Yuxuan Wang, Jianxin Liang, Jiansheng Wei, Huishuai Zhang, and Dongyan Zhao. VideoLLM knows when to speak: Enhancing time-sensitive video comprehension with video-text duet interaction format. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6338–6359. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.findings-emnlp.336.

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 (ICLR)*, 2026. URL https://openreview.net/forum?id=rxQnMSNCUs.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. StreamingVLM: Real-time understanding for infinite video streams. *arXiv preprint arXiv:2510.09608*, 2025.

George Yakushev, Nataliia Babina, Masoud Vahid Dastgerdi, Vyacheslav Zhdanovskiy, Denis Kuznedelev, Alina Shutova, and Max Ryabinin. Asynchronous reasoning: Training-free interactive thinking LLMs, 2025.

Weicai Yan, Yuhong Dai, Qi Ran, Haodong Li, Wang Lin, Tao Jin, Xing Xie, Hao Liao, and Jianxun Lian. Proact-VL: A proactive VideoLLM for real-time AI companions. In *International Conference on Machine Learning (ICML)*, 2026.

Haolin Yang, Feilong Tang, Lingxiao Zhao, Xinlin Zhuang, Yifan Lu, Xiang An, Ming Hu, Xiaofeng Zhang, Abdalla Swikir, Junjun He, Zongyuan Ge, Muhammad Haris Khan, and Imran Razzak. StreamAgent: Towards anticipatory agents for streaming video understanding, 2025a.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

Zhenyu Yang, Kairui Zhang, Yuhang Hu, Bing Wang, Shengsheng Qian, Bin Wen, Fan Yang, Tingting Gao, Weiming Dong, and Changsheng Xu. Livestar: Live streaming assistant for real-world online video understanding. In *Advances in Neural Information Processing Systems*, volume 38, 2025c. doi: 10.52202/085713-1050.

Zhenyu Yang, Kairui Zhang, Bing Wang, Shengsheng Qian, and Changsheng Xu. Livestarpro: Proactive streaming video understanding with hierarchical memory for long-horizon streams. *arXiv preprint arXiv:2606.17798*, 2026.

Dingyu Yao, Shuhuan Gu, Qingyi Si, Junhao Zhou, Chenxu Yang, Chuanyu Qin, Naibin Gu, Zheng Lin, Weiping Wang, Nan Duan, and Jiaqi Wang. Harnessing streaming video in the wild. *arXiv preprint arXiv:2606.08615*, 2026.

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, Lingpeng Kong, Qi Liu, Yuanxing Zhang, and Xu Sun. TimeChat-Online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM)*, pp. 10807–10816, 2025. doi: 10.1145/3746027.3754839.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. MiniCPM-V: A GPT-4V level MLLM on your phone. *arXiv preprint arXiv:2408.01800*, 2024.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 27890–27913, 2025.

Xueyang Yu, Cheng Shi, Yang Wang, and Sibei Yang. Eyes wide open: Ego proactive video-llm for streaming video. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0453.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *Conference on Language Modeling (COLM)*, 2024.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, Yi Wang, and Limin Wang. Streamforest: Efficient online video understanding with persistent event memory. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2547.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. Flash-VStream: Memory-based real-time understanding for long video streams, 2024. URL `https://arxiv.org/abs/2406.08085`.

Kairui Zhang, Zhenyu Yang, Bing Wang, Shengsheng Qian, and Changsheng Xu. Querystream: Advancing streaming video understanding with query-aware pruning and proactive response. In *International Conference on Learning Representations (ICLR)*, 2026.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *Transactions on Machine Learning Research (TMLR)*, 2025a.

Yichi Zhang, Xin Luna Dong, Zhaojiang Lin, Andrea Madotto, Anuj Kumar, Babak Damavandi, Joyce Chai, and Seungwhan Moon. Proactive assistant dialogue generation from streaming egocentric videos. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12044–12068. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.emnlp-main.605.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. LLaVA-Video: Video instruction tuning with synthetic data. *Transactions on Machine Learning Research (TMLR)*, 2025c.

Qi Zhao, Shijie Wang, Ce Zhang, Changcheng Fu, Minh Quan Do, Nakul Agarwal, Kwonjoon Lee, and Chen Sun. AntGPT: Can large language models help long-term action anticipation from videos? In *International Conference on Learning Representations (ICLR)*, 2024.

Ruixiang Zhao, Jie Yang, Zijie Xin, Tianyi Wang, Fengyun Rao, Jing Lyu, and Xirong Li. OmniPro: A comprehensive benchmark for omni-proactive streaming video understanding. *arXiv preprint arXiv:2605.18577*, 2026.

A Detailed Algorithms

Algorithm S1 PLAN: schema-walked generation of the control plan $\pi_t = (s_t, u_t)$

Input: causal snapshot C_t^{snap}; deferred question q; emitted-response history R_t; task threshold θ_{task}; optional-control threshold θ_{more}
Output: transient evidence state s_t; future-control update u_t; evidence confidence $p_{\text{hit},t}$

$P_t \leftarrow \textsc{Prompt}(q, R_t)$
$u_t \leftarrow \varnothing$

Current observation grounding
$z \leftarrow \textsc{Prefill}(C_t^{\text{snap}}, P_t \| \{\texttt{"seen":"}\})$
$(s_t.\text{seen}, z) \leftarrow \textsc{Decode}(z, \text{short text})$

Evidence sufficiency
$z \leftarrow \textsc{Prefill}(z, \texttt{","}, \texttt{"have_enough_info":})$
$p_{\text{hit},t} \leftarrow \frac{\exp z(\texttt{true})}{\exp z(\texttt{true}) + \exp z(\texttt{false})}$
$s_t.\text{hit} \leftarrow [p_{\text{hit},t} \geq \theta_{\text{task}}]$
$s_t.\text{event_time} \leftarrow \perp$; $s_t.\text{answer} \leftarrow \varnothing$
if s_t.hit **then**
  $z \leftarrow \textsc{Prefill}(z, \texttt{true,"event_time_s":})$
  $(s_t.\text{event_time}, z) \leftarrow \textsc{Decode}(z, \text{time})$
  $z \leftarrow \textsc{Prefill}(z, \texttt{,"answer":"})$
  $(s_t.\text{answer}, z) \leftarrow \textsc{Decode}(z, \text{short response})$
  $z \leftarrow \textsc{Prefill}(z, \texttt{","more":})$
else
  $z \leftarrow \textsc{Prefill}(z, \texttt{false,"more":})$
end if

Future-control update
$p_{\text{more}} \leftarrow \textsc{ReadProb}(z)$
if $p_{\text{more}} \geq \theta_{\text{more}}$ **then**
  $z \leftarrow \textsc{Prefill}(z, \texttt{true,})$
  $(\text{tail}, z) \leftarrow \textsc{Decode}(z, \text{optional control tail})$
  $u_t \leftarrow \textsc{ParseTail}(\text{tail})$
  ▷ u_t may update `next_check_s`, `question_for_next`, and `fps`
end if
return $(s_t, u_t, p_{\text{hit},t})$

Algorithm S2 SHOULDEMIT: temporally gated response decision

Input: transient evidence s_t; evidence confidence $p_{\text{hit},t}$; current video time t; gate state $G = (a, t_{\text{fire}})$; thresholds $\theta_{\text{hi}}, \theta_{\text{lo}}$; re-arm interval ρ; debounce interval d
Output: emission decision fire; updated gate state G

Re-arm the temporal gate
if $p_{\text{hit},t} < \theta_{\text{lo}} \ \vee \ t - t_{\text{fire}} \geq \rho$ **then**
  $a \leftarrow 1$
end if

Determine whether the current evidence defines a new event
candidate $\leftarrow [s_t.\text{answer} \neq \varnothing]$
fire $\leftarrow$ candidate $\wedge\, a \wedge [p_{\text{hit},t} \geq \theta_{\text{hi}}] \wedge [t - t_{\text{fire}} > d]$
if fire **then**
  $a \leftarrow 0$
  $t_{\text{fire}} \leftarrow t$
end if
$G \leftarrow (a, t_{\text{fire}})$
return (fire, G)

B EXPERIMENTAL IMPLEMENTATION DETAILS

B.1 COMPUTE INFRASTRUCTURE

All reported full-benchmark experiments were executed on NVIDIA GH200 GPUs on an HPC cluster.

We split each benchmark's official evaluation set at random into a 20% calibration split, used only to fit the response thresholds (Section 4.1), and an 80% held-out split on which every reported FORESIGHT number is computed.

Streaming evaluation is far more expensive than single-shot video question answering, because frames are ingested causally and the planner may run many times per video. A complete OmniPro evaluation of one configuration took about **150 GH200 GPU-hours**, covering ingestion, planning, response generation, and scoring.

In the reported results the visual sampling rate is set by the plan through its `fps` field, as listed in the Frames column of each table; the effect of removing this control is measured in Table 4.

B.2 IN-CONTEXT LEARNING AND EVALUATION PROMPTS

FORESIGHT trains nothing; it relies on two abilities the frozen VLM already has. Its world knowledge tells it how scenes tend to unfold, and in-context learning lets it follow the plan format from instructions in the prompt alone. Three examples show how the two combine.

- *Sports: "tell me when a goal is scored."* A player dribbles into the box. The model knows a shot may follow within seconds, so it brings `next_check_s` forward, raises `fps`, and sets `question_for_next` to "has the ball crossed the line?".
- *Cooking: "tell me when the water boils."* A pot has just been put on the stove. The model knows boiling takes minutes, not seconds, so it sets a long `next_check_s`, keeps `fps` low, and asks next time "are bubbles rising?".
- *Counting: "how many people enter the room?"* Someone walks towards the door. The model expects an entry soon and checks again shortly, and because one person has already been reported in R_t, it sets `new_event` only when a different person comes in, so the same entry is not counted twice.

In each case world knowledge says *when* something is likely to happen, and in-context learning turns that into a plan. No weights change.

For evaluation we use each benchmark's own task-specific prompt as provided, so that answers are scored under its official protocol. The prompt is prefilled once at the start of the stream and kept as a persistent prefix of the KV cache, and text written by the planner is never inserted into the visual history. We do not modify these prompts after seeing evaluation results, and the response thresholds are fitted only on the 20% calibration split.

C Additional OmniPro Analysis

C.1 Online-Mode Task Breakdown

Table 5 reports task-level Time F1, content accuracy and Joint F1. EA denotes Event-Alert; TG, Target-Ground; SM, State-Monitor; SC, Snapshot-Count; CA, Conditional-Alert; CC, Cumulative-Count; EN, Event-Narration; DC, Deduplicated-Count; and SI, Step-Instruction. Content correctness for Event-Narration and Step-Instruction is judged by GPT-4o-mini on the benchmark's five-level rubric, where a score of 3 or more counts as correct; all such responses were judged.

Dispider is evaluated on the 932-example audio-helpful/none subset supplied in its report, which gives no task-level content, so its rows are not directly comparable with the full-benchmark ones. MiniCPM-o 4.5 is our own run of the released checkpoint; for the same model and setting Zhao et al. (2026) publish per-task Joint F1 of 14.9, 8.7, 23.3, 16.0, 15.7, 7.6, 3.5, 27.3 and 7.5, for a mean of 13.8. StreamAgent's macro means are 12.5 (Time F1), 43.0 (content) and 5.0 (Joint F1), against the pooled values shown in the table.

Table 5: **OmniPro Online, per task (%).** The three blocks decompose the same emissions: Time F1 counts responses that land inside the ± 3 s window, content accuracy how many of those were right, and Joint F1 requires both. The two alert tasks are content-trivial once timed, so their content is 100 by construction. Time F1 $\times$ content $=$ Joint F1 holds for each task, but each source pools the Overall column differently, so it need not multiply out there. Dispider's content row is recovered from its other two rows task by task, so its Overall is their macro mean.

Method	EA	TG	SM	SC	CA	CC	EN	DC	SI	Overall
Time F1										
Dispider (Qian et al., 2025)	2.3	7.7	8.5	2.7	8.5	7.3	9.5	8.8	15.0	9.6
StreamAgent (Yang et al., 2025a)	6.8	8.8	10.5	9.0	11.3	5.7	16.0	17.2	27.3	15.0
QueryStream (Zhang et al., 2026)	8.2	6.9	24.9	7.5	17.6	22.1	19.9	21.1	30.9	17.7
MiniCPM-o 4.5 (Cui et al., 2026)	50.1	53.9	20.9	24.2	23.8	22.5	8.5	7.7	8.3	24.4
FORESIGHT	44.7	47.9	52.6	48.3	33.8	41.7	29.6	41.3	53.2	**43.7**
Content accuracy										
Dispider (Qian et al., 2025)	100.0	0.0	0.0	0.0	100.0	4.5	46.9	1.5	64.9	35.3
StreamAgent (Yang et al., 2025a)	100.0	16.1	37.9	21.2	100.0	15.4	23.2	46.9	26.2	36.2
QueryStream (Zhang et al., 2026)	100.0	19.6	12.4	13.1	100.0	29.7	36.4	33.5	26.8	41.3
MiniCPM-o 4.5 (Cui et al., 2026)	100.0	22.7	49.1	33.3	100.0	39.9	38.3	50.8	18.2	54.4
FORESIGHT	100.0	37.4	31.2	39.6	100.0	39.1	38.9	70.8	34.7	**54.6**
Joint F1										
LiveStar (Yang et al., 2025c)	9.7	0.8	0.0	0.0	14.7	0.0	1.6	0.0	6.0	3.6
MMDuet2 (Wang et al., 2026)	12.5	5.3	14.9	11.2	21.4	5.3	3.7	12.7	14.7	11.3
Dispider (Qian et al., 2025)	2.3	0.0	0.0	0.0	8.5	0.3	4.5	0.1	9.7	2.8
StreamAgent (Yang et al., 2025a)	6.8	1.4	4.0	1.9	11.3	0.9	3.7	8.1	7.1	5.0
QueryStream (Zhang et al., 2026)	8.2	1.4	3.1	1.0	17.6	6.6	7.2	7.1	8.3	6.7
MiniCPM-o 4.5 (Cui et al., 2026)	50.1	12.6	9.8	8.0	23.8	9.0	3.0	3.8	1.4	13.5
FORESIGHT	44.7	17.9	16.4	19.1	33.8	16.3	11.5	29.2	18.5	**23.0**

C.2 Emission Behaviour of the Baselines

How often a system speaks is worth separating from how well it is timed. Against the 9,051 ground-truth events, StreamAgent emits 0.79 responses per event and our MiniCPM-o 4.5 run only 0.37, yet MiniCPM-o 4.5 attains the higher Time F1 of the two (24.4 against 15.0). QueryStream sits at the opposite extreme at 2.75 responses per event, which lifts its Time F1 to 17.7 but, with most of those responses wrong, keeps its joint F1 low.

D Additional StreamingBench Results

D.1 Task-Level Breakdown

We split the original 18-task matrix into two readable tables. OP denotes Object Perception; CR, Causal Reasoning; CS, Clips Summarization; ATP, Attribute Perception; EU, Event Understanding; TR, Text-Rich Understanding; PR, Prospective Reasoning; SU, Spatial Understanding; ACP, Action Perception; and CT, Counting.

Table 6: **StreamingBench Real-Time Visual Understanding, per task (accuracy, %).** The first two blocks are reference points rather than streaming systems, taken from Lin et al. (2024b); a plain frame count means uniform sampling rather than a streaming rate. TimeChat-Online's keep-rate variants are from Yao et al. (2025) and the Qwen3-VL-8B and MiniCPM-o 4.5 rows from Lu et al. (2026).

Method	FPS	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	Avg.
Human	native	89.47	92.00	93.60	91.47	95.65	92.52	88.00	88.75	89.74	91.30	91.46
Gemini 1.5 Pro (Gemini Team et al., 2024)	1	79.02	80.47	83.54	79.67	80.00	84.74	77.78	64.23	71.95	48.70	75.69
GPT-4o (OpenAI, 2024)	64	77.11	80.47	83.91	76.47	70.19	83.80	66.67	62.19	69.12	49.22	73.28
Claude 3.5 Sonnet (Anthropic, 2024)	20	80.49	77.34	82.02	81.73	72.33	75.39	61.11	61.79	69.32	43.09	72.44
Video-LLaMA2-7B (Cheng et al., 2024)	32	55.86	55.47	57.41	58.17	52.80	43.61	39.81	42.68	45.61	35.23	49.52
VILA-1.5-8B (Lin et al., 2024a)	14	53.68	49.22	70.98	56.86	53.42	53.89	54.63	48.78	50.14	17.62	52.32
Video-CCAM-14B (Fei et al., 2024)	96	56.40	57.81	65.30	62.75	64.60	51.40	42.59	47.97	49.58	31.61	53.96
LongVA-7B (Zhang et al., 2025a)	128	70.03	63.28	61.20	70.92	62.73	59.50	61.11	53.66	54.67	34.72	59.96
InternVL2-8B (Chen et al., 2024c)	16	68.12	60.94	69.40	77.12	67.70	62.93	59.26	53.25	54.96	56.48	63.72
Kangaroo-7B (Liu et al., 2026a)	64	71.12	84.38	70.66	73.20	67.08	61.68	56.48	55.69	62.04	38.86	64.60
LLaVA-NeXT-Video-32B (**?**)	64	78.20	70.31	73.82	76.80	63.35	69.78	57.41	56.10	64.31	38.86	66.96
MiniCPM-V 2.6-8B (Yao et al., 2024)	32	71.93	71.09	77.92	75.82	64.60	65.73	70.37	56.10	62.32	53.37	67.44
LLaVA-OneVision-7B (Li et al., 2025a)	32	80.38	74.22	76.03	80.72	72.67	71.65	67.59	65.45	65.72	45.08	71.12
Qwen2.5-VL-7B (Bai et al., 2025b)	1	78.32	80.47	78.86	80.45	76.73	78.50	79.63	63.41	66.19	53.19	73.68
Qwen3-VL-8B-Instruct (Bai et al., 2025a)	–	80.10	76.60	81.10	81.10	77.00	73.20	80.60	63.40	68.80	57.00	74.10
VideoLLM-Online (Chen et al., 2024a)	2	39.07	40.06	34.49	31.05	45.96	32.40	31.48	34.16	42.49	27.89	35.99
Flash-VStream (Zhang et al., 2024)	1	25.89	43.57	24.91	23.87	27.33	13.08	18.52	25.20	23.87	48.70	23.23
Dispider (Qian et al., 2025)	1	74.92	75.53	74.10	73.08	74.44	59.92	76.14	62.91	62.16	45.80	67.63
TimeChat-Online (Yao et al., 2025)	1	80.22	82.03	79.50	83.33	76.10	78.50	78.70	64.63	69.60	57.98	75.36
55.8% keep rate	1	80.76	79.69	80.76	83.33	74.84	78.82	78.70	64.23	68.75	57.98	75.28
17.4% keep rate	1	79.13	81.25	78.86	80.77	70.44	77.26	77.78	67.07	66.19	53.72	73.64
StreamAgent (Yang et al., 2025a)	1	79.63	78.31	79.28	75.87	74.74	76.92	82.94	66.31	73.69	55.40	74.28
QueryStream (Zhang et al., 2026)	1	82.38	84.38	79.18	82.37	77.99	81.31	78.70	65.04	69.32	47.34	75.32
StreamBridge (Wang et al., 2025a), LLaVA-OV-7B	1	76.84	77.17	82.60	75.25	64.15	64.17	75.00	61.38	61.19	46.11	68.39
LLaVA-OV-7B + Stream-IT	1	82.29	72.44	92.09	80.86	71.07	74.46	75.00	62.20	70.26	28.50	70.92
StreamBridge (Wang et al., 2025a), Qwen2-VL-7B	1	80.38	78.74	83.22	79.86	74.21	69.47	77.78	63.41	69.97	43.01	72.01
Qwen2-VL-7B + Stream-IT	1	84.74	82.68	88.92	89.77	77.36	85.36	84.26	69.92	71.67	35.75	77.04
StreamForest (Zeng et al., 2025)	1	83.11	82.81	82.65	84.26	77.50	78.19	76.85	69.11	75.64	54.40	77.26
MiniCPM-o 4.5-9B (Cui et al., 2026)	–	80.90	88.30	82.00	85.00	78.90	85.40	78.70	67.10	75.60	56.00	78.20
Foresight	adaptive	83.7	76.8	90.5	88.6	79.1	85.0	84.3	75.9	75.5	43.2	79.30

ER denotes Emotion Recognition; SCU, Scene Understanding; SD, Source Discrimination; MA, Multimodal Alignment; ACU, Anomaly Context Understanding; MCU, Misleading Context Understanding; SQA, Sequential Question Answering; and PO, Proactive Output. The Contextual columns follow the benchmark's own order, which places ACU before MCU.

D.2 Proactive Output

Proactive Output is the one StreamingBench task that scores *when* a system speaks rather than what it chooses: a question counts as answered only if the response is produced within a time tolerance of the annotated moment. It is the task that tests the response gate most directly. Because it measures timing rather than answer accuracy like the other seventeen tasks, we keep it out of the Contextual average.

Table 7: **StreamingBench Omni-Source and Contextual, per task (accuracy, %).** These are the systems with results on the full 18-task evaluation; ‡ marks our own run under the official protocol. Baseline Contextual averages include Proactive Output, as in their sources. For FORESIGHT, the Contextual average and overall cover the three multiple-choice tasks, since Proactive Output scores timing rather than answer choice and is reported separately (46.3, 95% CI 39.1–53.7; Appendix D.2); averaged over all four tasks, its Contextual score is 44.50.

	Omni-Source					Contextual					
Method	**ER**	**SCU**	**SD**	**MA**	**Avg.**	**ACU**	**MCU**	**SQA**	**PO**	**Avg.**	**Overall**
Flash-VStream (Zhang et al., 2024)	25.91	24.90	25.60	28.40	26.00	24.80	25.20	26.80	1.96	24.12	24.04
VideoLLM-Online (Chen et al., 2024a)	31.20	26.51	24.10	32.00	28.45	24.19	29.20	30.80	3.92	26.55	32.48
Dispider (Qian et al., 2025)	35.46	25.26	38.57	43.34	35.66	39.62	27.65	34.80	25.34	33.61	53.12
StreamAgent (Yang et al., 2025a)	35.86	26.26	38.87	44.04	36.26	39.72	30.25	39.60	28.90	34.62	57.02
TimeChat-Online‡ (Yao et al., 2025)	36.62	26.75	38.55	45.90	36.96	41.45	32.27	40.55	28.25	35.63	58.11
StreamForest‡ (Zeng et al., 2025)	37.71	27.23	39.95	47.30	38.05	42.33	33.38	40.14	29.50	36.34	60.80
StreamBridge (Wang et al., 2025a), LLaVA-OV-7B	24.40	12.00	32.40	37.60	26.60	20.00	19.60	34.40	52.94	31.74	50.96
LLaVA-OV-7B + Stream-IT	20.80	13.20	43.60	27.60	26.30	20.40	17.60	41.60	37.26	29.21	51.73
StreamBridge (Wang et al., 2025a), Qwen2-VL-7B	32.00	15.20	39.60	38.40	31.30	25.20	21.20	33.20	66.67	36.57	55.09
Qwen2-VL-7B + Stream-IT	18.00	13.20	43.60	21.60	24.10	14.00	17.20	48.00	50.98	32.55	55.39
Qwen3-VL-8B-Instruct (Bai et al., 2025a)	44.40	27.20	46.40	49.60	41.90	30.00	38.40	38.00	52.80	39.80	59.30
MiniCPM-o 4.5 (Cui et al., 2026)	44.40	28.00	41.60	54.40	42.10	35.20	44.00	51.20	47.60	44.50	62.70
FORESIGHT	42.2	29.7	53.0	69.5	48.70	46.8	33.2	51.7	46.3	43.80	66.00

Our reported value, 46.3 (95% CI 39.1–53.7), comes from the run that follows the official protocol, with a median timing error of 4.0 s; the gate stays silent on 15 of its 175 questions. Letting the model see the full clip makes the gate fire on every question but leaves the score unchanged.

Against the other systems — Claude 3.5 Sonnet 64.7, GPT-4o 56.9, Gemini 1.5 Pro 45.1, and every open model at 40.9 or below — our gate is indistinguishable from Gemini and clearly behind the two strongest proprietary models. The confidence intervals are wide for every method because the task has few questions per model, so we treat Proactive Output as a diagnostic rather than as a headline result.

E ADDITIONAL OVO-BENCH ANALYSIS

To keep the main comparison legible, we split OVO-Bench into its Real-Time Visual Perception and temporal-reasoning task groups. OCR denotes Optical Character Recognition; ACR, Action Recognition; ATR, Attribute Recognition; STU, Spatial Understanding; FPD, Future Prediction; and OJR, Object Recognition. The row named LLaVA-Video-7B is the model listed as LLaVA-NeXT-Video-7B in the first release of the benchmark. The StreamBridge row is its Qwen2-VL-7B + Stream-IT configuration; its LLaVA-OV-7B backbone reaches 61.64 real-time and 48.13 overall here, and its StreamingBench results for both backbones are given in Tables 6 and 7. Human scores come from watching the video at its native frame rate (24–30 fps).

OVO-Bench has been revised since its first release, and the revision changed most baseline scores: Gemini 1.5 Pro, for instance, moves from 70.77/62.32/57.15/65.25 in v1 to 69.32/62.54/57.15/63.00 in v2, and Flash-VStream from 29.86 to 28.37 real-time average. Several method papers still reproduce the v1 leaderboard. All reference rows in the tables below are taken from the current v2 release of Niu et al. (2025) so that they are mutually consistent; rows attributed to a method paper keep that paper's own reported values. Note also that the TimeChat-Online row here is its 15.2% keep-rate configuration, whereas the StreamingBench tables use its 100% keep-rate configuration, because those are the configurations each benchmark's comparison reports.

E.1 REAL-TIME VISUAL PERCEPTION

Table 8: **OVO-Bench Real-Time Visual Perception, per task (accuracy, %).** The second block lists reference models rather than streaming systems, from the current (v2) release of Niu et al. (2025); Qwen3-VL-8B and MiniCPM-o 4.5 are from Lu et al. (2026).

Method	Frames	OCR	ACR	ATR	STU	FPD	OJR	Avg.
Human	native	94.0	92.6	94.8	92.7	91.1	94.0	93.20
Gemini 1.5 Pro (Gemini Team et al., 2024)	1 fps	85.91	66.97	79.31	58.43	63.37	61.96	69.32
GPT-4o (OpenAI, 2024)	64	69.80	64.22	71.55	51.12	70.30	59.78	64.46
LLaVA-Video-7B (Zhang et al., 2025c)	64	69.13	58.72	68.83	49.44	74.26	59.78	63.52
LLaVA-OneVision-7B (Li et al., 2025a)	64	66.44	57.80	73.28	53.37	71.29	61.96	64.02
Qwen2-VL-7B (Wang et al., 2024b)	64	60.40	50.46	56.03	47.19	66.34	55.43	55.98
InternVL2-8B (Chen et al., 2024c)	64	67.11	60.55	63.79	46.07	68.32	56.52	60.39
LongVU-7B (Shen et al., 2025)	1 fps	53.69	53.21	62.93	47.75	68.32	59.78	57.61
Qwen3-VL-8B-Instruct (Bai et al., 2025a)	1 fps	54.5	58.7	68.5	70.3	68.1	49.5	61.60
VideoLLM-Online (Chen et al., 2024a)	1 fps	8.05	23.85	12.07	14.04	45.54	21.20	20.79
Flash-VStream (Zhang et al., 2024)	1 fps	24.2	29.4	28.5	33.7	25.7	28.8	28.37
Dispider (Qian et al., 2025)	1 fps	57.7	49.5	62.1	44.9	61.4	51.6	54.55
TimeChat-Online (Yao et al., 2025)	1 fps	69.8	48.6	64.7	44.9	68.3	55.4	58.60
StreamForest (Zeng et al., 2025)	1 fps	68.46	53.21	71.55	47.75	65.35	60.87	61.20
StreamBridge (Wang et al., 2025a), Qwen2-VL-7B + Stream-IT	1 fps	84.56	71.56	74.14	49.44	75.25	72.83	71.30
QueryStream (Zhang et al., 2026)	1 fps	74.5	47.7	70.7	46.6	71.3	57.6	61.40
StreamAgent (Yang et al., 2025a)	1 fps	71.2	53.2	63.6	53.9	67.3	58.7	61.30
MiniCPM-o 4.5-9B (Cui et al., 2026)	1 fps	73.2	65.1	75.0	59.0	69.3	62.5	67.30
FORESIGHT	adaptive	85.91	71.27	75.88	64.97	78.81	73.25	75.02

E.2 BACKWARD TRACING AND FORWARD ACTIVE RESPONDING

EPM denotes Episodic Memory; ASI, Action Sequence Identification; HLD, Hallucination Detection; REC, Repetition Event Count; SSR, Sequential Steps Recognition; and CRR, Clues Reveal Responding.

Table 9: **OVO-Bench temporal-reasoning breakdown (accuracy, %).** Reference rows follow the same provenance as Table 8, except VideoLLM-Online's Forward Active Responding and overall, which the original release does not report and which come from our own run.

	Backward Tracing				Forward Active Responding				
Method	**EPM**	**ASI**	**HLD**	**Avg.**	**REC**	**SSR**	**CRR**	**Avg.**	**Overall**
Human	92.6	93.0	91.4	92.30	95.5	89.7	93.6	92.90	92.80
Gemini 1.5 Pro (Gemini Team et al., 2024)	58.59	76.35	52.64	62.54	35.53	74.24	61.67	57.15	63.00
GPT-4o (OpenAI, 2024)	57.91	75.68	48.66	60.75	27.58	73.21	59.40	53.40	59.54
LLaVA-Video-7B (Zhang et al., 2025c)	56.23	57.43	7.53	40.40	34.10	69.95	60.42	54.82	52.91
LLaVA-OneVision-7B (Li et al., 2025a)	54.21	55.41	21.51	43.71	25.64	67.09	58.75	50.50	52.74
Qwen2-VL-7B (Wang et al., 2024b)	47.81	35.48	56.08	46.46	31.66	65.82	48.75	48.74	50.39
InternVL2-8B (Chen et al., 2024c)	48.15	57.43	24.73	43.44	26.50	59.14	54.14	46.60	50.15
LongVU-7B (Shen et al., 2025)	40.74	59.46	4.84	35.01	12.18	69.48	60.83	47.50	46.71
Qwen3-VL-8B-Instruct (Bai et al., 2025a)	58.8	41.1	23.1	41.00	21.2	51.3	40.4	37.70	46.80
VideoLLM-Online (Chen et al., 2024a)	22.22	18.80	12.18	17.73	11.20	44.47	38.80	29.00	21.12
Flash-VStream (Zhang et al., 2024)	39.1	37.2	5.9	27.38	8.0	67.3	60.0	45.09	33.61
Dispider (Qian et al., 2025)	48.5	55.4	4.3	36.06	18.1	37.4	48.8	34.72	41.78
TimeChat-Online (Yao et al., 2025)	53.9	62.8	9.1	42.00	32.5	36.5	40.0	36.40	45.60
StreamForest (Zeng et al., 2025)	58.92	64.86	32.26	52.02	32.81	70.59	57.08	53.49	55.57
StreamBridge (Wang et al., 2025a), Qwen2-VL-7B + Stream-IT	67.68	57.43	79.03	68.05	19.17	64.25	61.67	48.36	62.57
QueryStream (Zhang et al., 2026)	54.2	63.5	8.6	42.10	33.2	43.1	40.8	39.03	47.51
StreamAgent (Yang et al., 2025a)	54.8	58.1	25.8	41.70	35.9	48.5	52.0	45.40	49.40
MiniCPM-o 4.5-9B (Cui et al., 2026)	55.2	66.9	45.7	55.90	43.0	66.0	58.8	55.90	59.70
FORESIGHT	62.2	69.5	33.6	55.10	32.1	77.2	59.8	56.37	62.16

F ANTICIPATION ANALYSIS

This experiment is what led us to FORESIGHT. We asked a simple question: if a frozen VLM watches a stream and is stopped T seconds before an event, can it already tell that the event is coming? If it can, the model does not need to be asked about every frame. It can decide for itself when the next moment worth looking at will arrive, and spend its computation there. Figure 4 measures this: at each lead time T, the model sees the stream up to T seconds before an event and must anticipate it from that context alone.

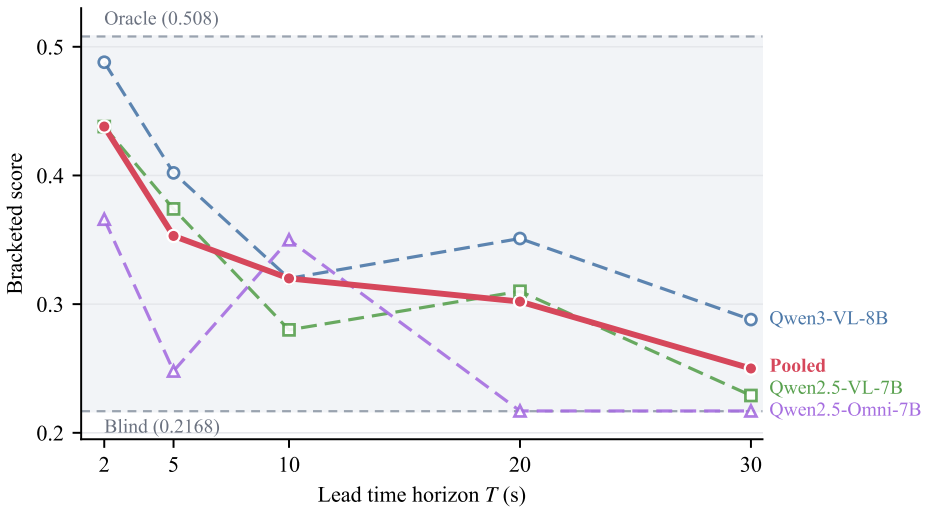


Figure 4: **Anticipation quality against lead time.** Each curve shows how well a frozen VLM anticipates an upcoming event when it is asked T seconds before the event, using only the stream seen so far. Scores are bracketed between two references: *Blind* (0.2168), a baseline that never sees the video, and *Oracle* (0.508), which observes the event itself. Dashed lines are individual backbones and the solid red line pools them. The pooled score is well above the blind baseline for near-future events and falls steadily as the horizon grows, approaching it by $T = 30$ s; individual backbones are noisier, and Qwen2.5-Omni-7B reaches the blind baseline from $T = 20$ s. Anticipation is therefore most reliable over the short horizons at which FORESIGHT schedules its next check.

Anticipation is real, but short-lived. The answer is yes, for the near future. Two seconds before an event, the pooled score closes about three quarters of the gap between the blind baseline and the oracle, so the model already sees much of what is about to happen. By 30 s that share has fallen to about a tenth. This matches intuition: a player dribbling into the box makes a shot in the next few seconds likely, but says little about the score half a minute later. What a model can infer about the future comes from cues that are visible now, and those cues stop being informative as the horizon grows.

Why the controller must check often. This decay shapes how FORESIGHT plans. Each plan is a short bet on the immediate future: it says when to look again, what to look for, and how densely to sample until then. Because the model's view of the future is reliable only a few seconds ahead, those bets must be renewed often. A controller that checked once and scheduled the next check far ahead would be acting on a prediction it can no longer support, and would miss events that its last look could not foresee. FORESIGHT therefore keeps checking at a steady, scene-dependent rate, lengthening Δ_t when nothing is developing and shortening it when something is. Keeping those checks cheap is the subject of Appendix G.

Fewer checks is an open problem. The same result marks the limit of our approach. The ideal controller would check about once per event, and a system that could plan reliably tens of seconds ahead would come close. Current VLMs cannot: their foresight fades within seconds, so a controller built on them must spend some checks on moments where nothing happens. Reaching the ideal rate

of one check per event, or even a few checks per minute on calm streams, would need a model whose anticipation lasts much longer than today's.

Training for anticipation. We see this as a property of how VLMs are trained rather than a fixed limit. Pretraining teaches a model to describe what is in front of it, not to predict what comes next, so any foresight it has is a by-product. An objective that asks the model to predict upcoming events, their timing, or the next useful moment to look would make anticipation part of what the model learns directly. Such a model would be proactive by nature, and a controller like FORESIGHT could then plan further ahead with fewer checks.

G INFERENCE EFFICIENCY

FORESIGHT is designed so that thinking about the stream costs little beyond watching it. We make this precise with a simple cost model and show where each design choice enters it.

Setup. Consider a stream of length L seconds in which N frames $x_1, \dots, x_N$ are admitted, each encoded into m visual tokens, and which contains E events of interest. The Think LLM runs at check times $t_1 < \cdots < t_K$ with $t_{k+1} = t_k + \Delta_{t_k}$, so its average check rate is $q = K/L$. We measure work $\mathcal{W}$ in tokens processed by the VLM; with P parameters, a dense forward pass costs about $2P$ FLOPs per token, so compute is $2P\,\mathcal{W}$.

The baseline: re-reading the stream. A controller without a shared cache must rebuild its context at every check. At check k it prefills the whole state C_{t_k}, which holds about $m\,n_k$ tokens for the n_k frames seen so far, plus the prompt of length $|p|$, before writing a plan of $|\pi|$ tokens:

$$\mathcal{W}_{\text{re-read}} \;=\; \sum_{k=1}^{K} \big(m\,n_k + |p| + |\pi|\big) \;=\; \mathcal{O}(K\,m\,N)\,. \tag{3}$$

Every check pays again for the whole history. With evenly spaced checks $K \propto L$ and $N \propto L$, so this cost grows as L^2: doubling the stream quadruples the work.

FORESIGHT. FORESIGHT splits the same work into what must be paid once and what is paid per decision:

$$\mathcal{W}_{\text{FORESIGHT}} \;=\; \underbrace{\rho\,m\,N_{\text{adapt}}}_{\text{ingest, once}} \;+\; \underbrace{K\,c_{\text{probe}}}_{\text{checks}} \;+\; \underbrace{\textstyle\sum_j |u_{t_j}| + |s_{t_j}|}_{\text{plans at evidence}}, \tag{4}$$

where the last sum runs over the steps j at which the probe finds enough evidence to write a plan. Each term corresponds to one design choice.

- *Shared KV cache.* Each visual token is prefilled exactly once, when the Ingest LLM appends it to C_t. The Think LLM reads a snapshot $C_{t_k}^{\text{snap}}$ of that cache instead of rebuilding it, so the $m\,n_k$ term of Eq. 3 disappears from every check, and the prompt p, prefilled once as a persistent prefix, disappears with it.
- *Adaptive frames and pruning.* The plan admits $N_{\text{adapt}} \le N$ frames, sampling densely only around expected events, and Keep and Ignore Classes retain a fraction $\rho \le 1$ of each frame's tokens. Both shrink the one-time ingest term, which is the largest in Eq. 4.
- *The probe inside the plan.* Most checks find nothing new. For those, the Think LLM decodes only the first field of the plan and reads `have_enough_info` from a single step of logits, so a check costs a constant c_{probe} of a few tokens rather than a full plan.
- *Diff-based plans.* When evidence does arrive, the model writes only the fields that change, $|u_{t_j}| + |s_{t_j}| \ll |\pi|$, and the merge $U \leftarrow \text{merge}(U, u_t)$ keeps the rest.

What this buys. Two properties follow. First, the controller's cost grows with the number of decisions, not with the length of the video: the check term is linear in K with a small constant, and the plan term is roughly proportional to the number of events E. Second, the cost of each decision does not grow over time. A decoded token attends to the cache, so its cost scales with $|C_t|$; compaction and eviction keep $|C_t|$ within a fixed budget B, so planning at the end of a long stream is as cheap as at the start. The one-time ingest is the cost of seeing the video at all, and no controller can avoid it.

A natural lower bound is the *oracle*, a controller with the same ingest that knows in advance when the E events occur and plans exactly once at each of them:

$$\mathcal{W}_{\text{oracle}} \;=\; \rho\,m\,N_{\text{adapt}} \;+\; \sum_{j=1}^{E} |u_{t_j}| + |s_{t_j}|. \tag{5}$$

Subtracting Eq. 5 from Eq. 4 leaves only the price of not knowing the future,

$$\mathcal{W}_{\text{FORESIGHT}} - \mathcal{W}_{\text{oracle}} \;\approx\; K\,c_{\text{probe}}, \tag{6}$$

a few tokens per check. FORESIGHT keeps this term small in two ways: c_{probe} is tiny, and K adapts to the video, because the plan lengthens Δ_t when nothing relevant is happening.

Why the design choices matter. Each saving above is a design choice, and giving up any one of them is costly. Under the cost model, at one check every four seconds, dropping the shared KV cache raises total compute about $40\times$, since every check re-reads the stream; sampling at a fixed rate without pruning roughly triples it by inflating the ingest term. The other choices govern the controller's overhead above the oracle: a full plan instead of a diff makes it about $25\times$ larger, a diff at every check without a probe about $4\times$, and a probe run as a separate pass about $2\times$. These factors come from the model, not from measurement, but their ordering follows from the structure of Eq. 4.

Where FORESIGHT operates. Figures 5 and 6 plot the cost model against the check rate $q = 1/\Delta_t$, the number of planning steps per second of video. The oracle of Eq. 5 checks once per event, so its rate $q = E/L$ is the *ideal operating point*. No real controller can sit there, because knowing when an event will happen is exactly what a streaming model has to work out. FORESIGHT instead chooses Δ_t from what it sees, so its check rate is a property of the video rather than a fixed setting: calm, slowly changing scenes let it wait several seconds, while fast motion, many objects or a quickly developing goal pull the next check closer. We therefore draw its operating point as a distribution: most streams fall in a window of one check every 2 to 10 s, and event-dense streams reach one check per second.

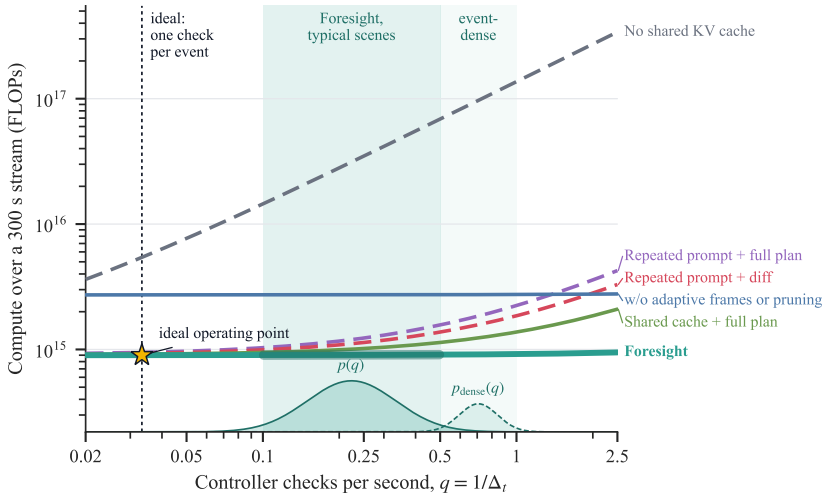


Figure 5: **Total compute against check rate.** Each curve is the compute a design spends on a 300 s stream. Designs that re-read the stream at every check grow quickly with q: without a shared KV cache every check prefills the whole history again, and with a repeated prompt every check pays for the prompt on top of the plan. Ingesting at a fixed rate without pruning costs the same at every q, so it appears as a flat line set by how many frames are admitted. FORESIGHT stays almost flat across its operating region: nearly all of its cost is the one-time ingest of the video, and each check adds only a short probe. The shaded bands and the distributions $p(q)$ and $p_{\text{dense}}(q)$ show where its check rate falls in typical and event-dense scenes; the star marks the ideal operating point of one check per event.

Together the figures make one point from two sides: in total compute FORESIGHT is barely distinguishable from the oracle, and in the extra compute it does spend it is the cheapest design we consider. The curves use representative token counts, and the distributions $p(q)$ and $p_{\text{dense}}(q)$ are assumed; measured check rates on the benchmarks would replace them.

G.1 MEASURED COST OF THE STREAMING BASELINES

Online task scores say nothing about whether a system can keep pace with the stream it is watching. The natural reference is the video itself: a real-time system must consume one second of footage in at most one second of compute. We therefore record, for the baselines we ran ourselves, the real-time (RT) factor, defined as wall-clock seconds of compute per second of video. All measurements use a single GH200 per sample on the same OmniPro Online set (2,700 videos, 143 hours of footage),

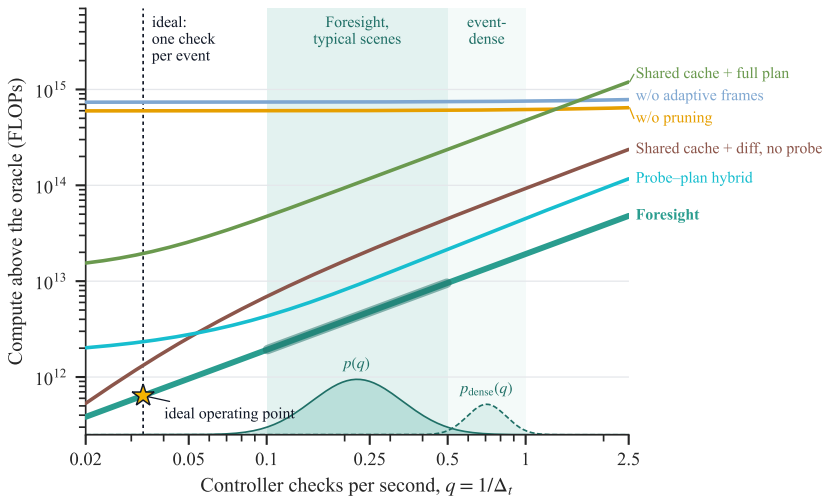


Figure 6: **Compute above the oracle.** The same model, with the oracle's compute subtracted from every design. Because the oracle shares the ingest pipeline, what remains is only the work a design spends beyond the least it could, which makes differences that are invisible in Figure 5 easy to read. Removing adaptive frames or pruning adds a fixed cost from extra visual tokens, so these curves are flat. The controller variants grow with q, and their order shows what each saving buys: a full plan at every check costs most, then a diff at every check, then a separate probe followed by a plan, and least of all FORESIGHT, which folds the probe into the plan and writes a diff only when evidence arrives. In its operating region FORESIGHT wastes about half as much as the probe–plan hybrid, a quarter as much as a diff at every check, and about a hundredth as much as the designs without adaptive frames or pruning.

and the wall clock includes the shared per-frame decoding and vision-tower cost attributed back to each sample.

Neither StreamAgent arm runs in real time. With every mechanism of the method active, each second of video costs roughly two to three seconds of GH200 time — 2.13× with the omni backbone and 2.70× with the vision-only one — so on this hardware it falls behind the stream. The gap is structural, since every clip triggers a caption, two candidate plans, a plan-scoring pass and a trigger check before any answer is produced. The probe configuration, which drops per-clip planning, plan scoring and the trigger gate and keeps only captioning plus two answerer calls per ground-truth trigger, runs at 0.58× – faster than the stream, at 26% of the online cost.

QueryStream sits at the opposite end of the same axis. Measured over every scored OmniPro sample, it averages 0.053× on its Qwen2.5-VL-7B backbone and 0.084× on Qwen3-VL-8B, so both finish well inside real time, and the newer backbone is consistently the more expensive of the two. As for StreamAgent, the ratio is close to flat across video-length buckets, confirming that cost scales with duration rather than with anything specific to long videos.

H CONTROLLER CONFIGURATION AND CODE RELEASE

Table 10 lists the response-gate settings used on OmniPro Online, fitted per task on the 20% calibration split and held fixed on the 80% evaluation split.

Table 10: **Response-gate settings on OmniPro Online.** Threshold on $p_{\mathrm{hit},t}$, gate mode, and refractory interval for each task, fitted on the calibration split.

Task	Abbr.	Threshold	Mode	Refractory (s)
Target-Ground	TG	0.50	edge	300
Event-Alert	EA	0.45	edge	600
State-Monitor	SM	0.80	level	7
Conditional-Alert	CA	0.98	level	10
Step-Instruction	SI	0.01	level	7
Snapshot-Count	SC	0.985	edge	600

Each setting has three parts. The *threshold* is the evidence confidence $p_{\mathrm{hit},t}$ at which the gate may fire. The *mode* decides how often it fires: an *edge* gate fires once when the confidence first crosses the threshold, and a *level* gate fires whenever the confidence stays above it. The *refractory interval* is the shortest time, in seconds, allowed between two responses, which stops a lasting event from being reported again. These values come from the video-only configuration used for all reported results. The settings follow the nature of each task: alerts and counts that should be reported once per event use an edge gate with a long refractory interval, while tasks that track a changing state, such as State-Monitor and Step-Instruction, use a level gate with a short one so that every new step or state change can be reported.

Code and materials. The code, prompts, fitted controller settings, and evaluation scripts used in this paper will be released publicly on GitHub upon publication.