JINGDONG

JoyAI-VL-Interaction: Real-Time Vision-Language Interaction Intelligence

Dingyu Yao*, **Junhao Zhou***, **Chenxu Yang***, **Chuanyu Qin***, **Haowen Hou***, **Zheming Liang**, **Congcong Wang**, **Yuhang Cao**, **Shenglong Ye**, **Shuai Xie**, **Shuhuan Gu**, **Haoyang Huang**, **Qingyi Si***,†, **Nan Duan**✉, **Jiaqi Wang**✉

JD.com

*Equal Contribution, †Project Lead, ✉Corresponding Authors

Abstract

Many moments in the real world do not wait for a user to ask. A fire starts on a security monitor, an expression flickers across a video call, or a product a viewer wants flashes by in a livestream. Once missed, the moment is gone, because the physical world does not pause. Yet today's large models remain mostly turn-based by design: they answer only when addressed, and even video-call apps that appear interactive still operate as question-answer systems, reacting only when polled or prompted. We argue for a different paradigm: a model that is present in the world like a person. It continuously watches what is happening now, decides on its own whether to speak or stay silent, interacts in real time, and delegates to a background model when the problem is hard. This opens a "watch-and-do" mode of human-AI collaboration, expands AI assistance to far more real-life situations, and aligns naturally with the real-time demands of embodied intelligence. To advance interaction models and their adoption across domains, we make two fully open-sourced contributions. First, we release **JoyAI-VL-Interaction**, an 8B-scale, vision-first VL-interaction model that is proactive by design. The model makes the response decision internally, choosing each second to stay silent, respond, or delegate to a background model, and it excels at vision-triggered responsiveness and time awareness. We pair it with a transferable training recipe, from which capabilities we never trained for emerge, such as guiding a shopper through changing app screens or improvising a lecture from a slide deck. Second, we release a complete, deployable system built around that model. The system streams any ongoing video, from cameras, livestreams, or security feeds, into the model, making it genuinely present in the world. It supports hours of continuous video with sub-second latency. All other components are pluggable, including ASR/TTS modules, memory, visualization UI, and a background brain that can connect to any API, model, or agent. Across six real-world streaming scenarios, human raters prefer JoyAI-VL-Interaction over the in-app video-call assistants of Doubao and Gemini by a wide margin on both quality and timing, winning 77.6% against Doubao and 87.9% against Gemini. To our knowledge, this is the first open, vision-driven interaction model released together with its training recipe, data, and complete deployable system. With our repository, anyone can deploy an assistant that sees the present, speaks up on its own, and moves fluidly between the physical and digital worlds. Our goal is to move VLMs beyond turn-based dialogue toward real streaming interaction, openly and together.

Project page: https://joyai-vl-video-future-academy-jd.github.io/JoyAI-VL-Interaction/
Repository: https://github.com/jd-opensource/JoyAI-VL-Interaction
Release: The model weights, interaction data, and full system code will be released by June 20, 2026.

1 Introduction

The moments that most need AI are often the very moments no one has time to ask for it. A toddler wanders toward a hot stove. The decisive play in a match is over before anyone has time to cheer. An aging parent living alone falls in the next room. What these moments share is not merely urgency, but timing: the instant that calls for a word, a cheer, or a steadying hand arrives before anyone thinks to ask, and disappears just as quickly. Today's large models are not built for such moments. Most remain fundamentally turn-based: they wait to be addressed, and respond only after they are asked. This limitation is structural, not merely a matter of speed. A turn-based model cannot perceive "now" by construction: it opens its eyes only at the moment it is prompted. What these situations require instead is a model that continuously sees what is happening in the present and interacts with the world as events unfold.

Existing efforts do not truly solve this problem. One line of work is real-time omni models, such as GPT-Realtime-2 [17] and Qwen3.5-Omni [22], which are genuinely end-to-end single models (audio and video in; speech out) with low latency and no stitched-together pipeline. However, their optimization target remains conversational turn-taking: responding as quickly as possible after the user has spoken. Fundamentally, they are still organized around dialogue. They wait for the user's turn, and are not designed to continuously watch a visual stream and decide on their own when the moment calls for speaking. Another line of work is consumer video-call products. Doubao's in-app video call [2] chases proactivity by firing a background request every few seconds, but its core is still turn-based and its autonomy is locked to the polling cycle: an event is not observed until the next poll, so the system can never react sooner than that interval. Gemini's in-app video call [7] goes even less far: it does not perform the background polling that Doubao relies on to monitor for events, and instead remains strictly in a one-question, one-answer interaction pattern. A third line is research on streaming-video understanding [3, 8, 19, 20, 26, 29, 33], which mostly remains at the laboratory stage. It is often evaluated in the manner of offline video understanding [12, 13], and it rarely brings together the properties a real-world deployment demands [31], namely proactive response[19, 27, 29], long-horizon memory [32, 34, 35], and real-time operation [25, 28]. It typically advances only one dimension at a time. In short, none of these systems meets the needs of real deployment: each is either turn-driven rather than event-driven, not genuinely present in the live world, or alive only inside a benchmark.

JoyAI-VL-Interaction is built for exactly this. We adopt the term *interaction model* [11] from Thinking Machines Lab (TML) and make its criterion explicit: an interaction model judges for itself, moment to moment, when the time is right to respond to the ongoing stream, the way a person does, choosing when to speak and when to stay silent rather than answering only when addressed. A turn-based model, however low its latency or native its architecture, cannot choose its own moment: it speaks only when a user's turn arrives. The difference is not how fast a model answers, but whether it decides for itself when answering is worth it. We and TML recognized at nearly the same time that interactivity is worth scaling as a capability of its own; where we part is the route to it. TML fuses speech and vision into one large model and shared a research preview; we make vision the first-class driver with speech as pluggable I/O, keep the model compact enough to run on local, low-cost hardware, and open the whole stack so anyone can deploy, reproduce, and build on it.

Contribution 1 — the VL-interaction model. Our central idea is simple: make when to act a learned, per-second decision of the model itself, with staying silent treated as a first-class action alongside speaking and delegating. We build this vision-first interaction model on our base vision-language model, JoyAI-VL 1.0 (§3.1). Speaking and staying silent are the root of proactivity and time awareness, knowing when a moment is worth a word and when it is not; delegating is how the model reaches for more capability when a problem outgrows real-time inference. We pair every second of the visual stream with the action it calls for, yielding time-aligned data, which we construct at scale (§3.2); from this the model learns its interactivity, as a behavior of its own rather than something bolted on by external turn-detectors or activity heuristics. To keep it affordable over an unbounded stream, we encode the video with AdaCodec[9], which spends far fewer tokens on each predictable frame so the budget grows far more slowly over a long stream (§3.1). We open-source the model weights, the training recipe, and the data behind it.

Contribution 2 — the VL-interaction system. We open-source a complete system that works out of the box: anyone can feed it a webcam or a livestream, and the interaction model immediately sees what is happening and interacts with the user in real time, genuinely present in the scene. Around the model, the

Figure 1 JoyAI-VL-Interaction processes a continuous video stream in real time, deciding at each step whether to respond, stay silent, or delegate complex tasks to a background model for asynchronous handling.

system provides everything a deployment needs: ASR and TTS, a visualization UI, long-horizon memory, and a background bridge to any API, model, or agent the user brings, all served on standard vLLM to stay real-time over hours of continuous video. Among these, the interaction model alone decides whether and when to interact; the rest only transduce and orchestrate around it. Each ships with an open-source default, and the system provides the hooks to replace it, so a deployment can drop in its own ASR/TTS API or custom modules without rebuilding the stack. The system runs two concurrent loops, joined by the model's "delegate" action: a real-time loop with the user, and an asynchronous loop in which the model offloads a hard problem to a background brain while staying present with the user, folding the result back in when it returns. In a head-to-head human evaluation across six everyday scenarios, JoyAI-VL-Interaction is preferred over both Doubao's and Gemini's in-app video-call assistants by a wide margin on quality and timing, winning 77.6% of comparisons against Doubao and 87.9% against Gemini. Its strongest margins fall on the most time-critical settings: it wins every comparison in monitoring and alerting against both systems and never loses one in real-time translation or counting, exactly the event-driven, act-at-the-right-moment tasks that turn-based products structurally miss (§5).

The value here is more than a better "video assistant." When a model can be present and decide for itself what to watch and when to speak, human–AI collaboration shifts from "submit a request and wait" to "watch-and-do," and AI can step into far more of everyday life, honoring embodied intelligence's premise that the physical world cannot be paused. The same capability is the substrate for a class of long-awaited products: a companion that is truly present, AI glasses that translate or caption what you see, an aide that serves as eyes for those who cannot, and uses no one has imagined yet. We release JoyAI-VL-Interaction, the model, its training recipe and data, and the full system in the open, precisely so the community can build and explore all of this, and more, together. Our larger hope is to move the field from turn-based dialogue toward genuine streaming interaction, in the open.

2 Related Work

2.1 Turn-based Models and Products

Almost all of today's models are *turn-based*: they act only when a user takes a turn, answering once addressed and otherwise idle. This is a structural property, not a matter of speed. A turn-based system has no mechanism for reacting to an event in the world that arrives with no accompanying user utterance, however quickly it can reply once spoken to. Two families of recent systems push hard on responsiveness while keeping this turn-based core.

Realtime and omni models. OpenAI's GPT-Realtime-2 reasons inside a single speech-to-speech loop [17], and Alibaba's Qwen3.5-Omni [22] is a natively pretrained omni-modal model (text, audio, image, video) with real-time streaming and built-in turn-taking and interruption handling, the latter openly released. These remove the latency of cascaded stacks, but what they optimize is *conversational turn-taking*: how quickly and naturally they answer once the user has spoken. Their interaction is organized around dialogue and waits for the user's turn; they are not built to watch a visual stream and decide, unprompted, that this is the moment to speak. Being open, real-time, and multimodal (as Qwen3.5-Omni already is) is therefore necessary but not sufficient for the setting we target.

Consumer video-call products. In-app "video call" features are the most familiar real-time multimodal assistants and our most recognizable baseline, yet their cores remain turn-based. Doubao's runs on ByteDance's Volcano Engine conversational-AI stack, a configurable ASR/VLM/TTS pipeline with stop/turn detection, and its documentation makes the mechanism plain: while no one is speaking, the server only caches incoming video frames and does not send them to the model, so the application must periodically fire an `ExternalTextToLLM` trigger to obtain any analysis [24]; this characterization is based on the publicly available version of the stack, which may differ from the deployed product. "Monitoring" is thus an external clock bolted onto a turn-based model: reaction to an on-screen event waits for the next trigger and can never beat the polling interval. Gemini's video call goes less far still; in our use it answers only about the frame visible the instant a question is asked. In neither case does the decision of when to respond live in the model.

2.2 Interaction Models

A distinct, newer line breaks with the turn-based assumption by moving interactivity into the model.

TML interaction models. Thinking Machines Lab recently named this an *interaction model* and argued, as we independently and concurrently concluded, that scalable interactivity should come from the model itself rather than an external harness of turn detectors and activity heuristics. Their model handles audio, video, and text in real time and can speak unprompted, acting on the ongoing stream rather than waiting for a user's turn, and, as in our system, it is paired with an asynchronous background model for slower, harder reasoning, the two sharing context; it was released as a research preview [11]. That two groups arrived at this approach at nearly the same moment we read as a sign that the move from turn-based to interactive models is a direction whose time has come. Within this shared interaction–background design, our aesthetic differs in two deliberate ways. On modality, rather than fusing audio and vision into the model alike, we keep speech (ASR/TTS) a pluggable module and make vision the model's intrinsic, primary modality for proactive interaction, fitting a watch-and-interact setting in which voice is interchangeable I/O. On scale, TML-Interaction-Small is a 276B-parameter mixture-of-experts model (12B active) that its authors describe as the smallest they can serve at the required latency, whereas we deliberately choose a compact ∼8B model (balancing interaction-grade responsiveness, local and low-cost deployability, and capability) so that it can be run, fine-tuned, and reproduced widely rather than only on large infrastructure.

Full-duplex speech. For speech, Kyutai's MoshiRAG [4] is the prominent open instance: a real-time, full-duplex speech–text model in which both sides can speak and listen at once rather than strictly alternating, and the closest open precedent for putting interaction in the model. Its modality and scope (spoken conversation) set it apart from the visual, event-driven setting we address.

2.3 Streaming Video Understanding

A research line lets models process video as it arrives rather than after the fact, spanning streaming video LLMs, proactive response [19, 27, 29], real-time inference [25, 28], and long-horizon video memory [32, 34, 35]. It is closest to our model contribution, but three gaps separate it from a deployable interaction model. It typically advances one property at a time (responsiveness, *or* proactivity, *or* memory) rather than the three together. It is usually measured on offline benchmarks, at times even in the manner of offline video understanding, which never tests reaction to live events under real-time constraints. And it stops at the model, without the surrounding system (serving, memory, transduction, delegation) that hours of sustained real-time presence demand.

2.4 Our Position.

Read along two axes, *how* a model interacts and *what* is released, JoyAI-VL-Interaction occupies a cell no prior work does. Against turn-based systems, native or product, the decision of when to act lives inside our model and is taken every second, event-driven, so reaction is bounded by inference rather than by a user's turn or a trigger clock. Against TML's interaction model, we make vision the first-class driver, the watch-and-interact setting rather than audio–video conversation, and we open-source not a research preview but the model, its training recipe, and a complete, composable, deployable system. Against MoshiRAG, we are vision-driven rather than speech-centric, reacting to events that carry no conversational turn at all. And against streaming-video research, we bring responsiveness, proactivity, and memory together in one model and run it inside a system built for sustained presence. To our knowledge, JoyAI-VL-Interaction is the first open, vision-driven interaction model released together with a complete deployable system: the point where all of these properties meet, rather than any one alone.

3 JoyAI-VL-Interaction Model

Figure 1 summarizes the model. JoyAI-VL-Interaction watches a streaming visual input and, at every second, takes one of three actions: speak to the user, stay silent and keep watching, or delegate a slower, harder task to an asynchronous background model whose result is folded back into the stream when ready. The first two actions form a real-time loop with the user; the third, an asynchronous loop with the background.

Two design points frame the rest of this section. First, the model learns a background-agnostic delegation protocol, so the background can be swapped for any external model, agent, or API; we cover the model side here and the system side in §4. Second, the model does not process or generate speech itself: turning voice into text and back is left to pluggable ASR/TTS in the system, keeping the autonomous, vision-driven core decoupled from interchangeable I/O.

The rest of the section builds the model from this behavior outward: the base model and architecture (§3.1), the data that teaches when to speak, stay silent, and delegate (§3.2), and the training recipe (§3.3), which covers continue training, the training objective, reinforcement learning, and infrastructure.

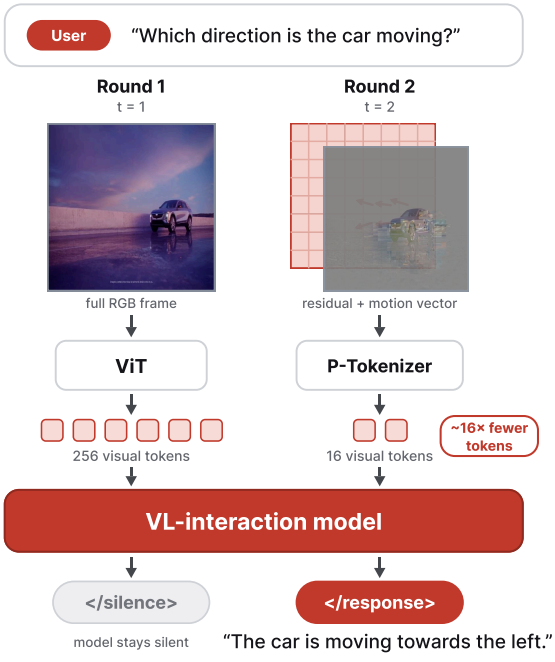

Figure 2 Model overview and video encoding with AdaCodec [9].

3.1 JoyAI-VL 1.0 and Architecture

JoyAI-VL 1.0. JoyAI-VL-Interaction is built on JoyAI-VL-1.0, where the language model is initialized from Qwen3-8B [30], the visual encoder is the Qwen3-VL ViT [1], and the projection layer between them is trained from scratch. JoyAI-VL 1.0 is obtained through three stages,

representation alignment, vision-language pre-training and post-train with On-Policy Distillation [15] and RL [14, 23]. At this point the model is a conventional, turn-based VLM. Its per-second interaction behavior, deciding each second whether to speak, stay silent, or delegate, is acquired afterward through the interaction-training recipe in §3.3.

Native Streaming Video Codec. We tokenize the video stream with AdaCodec [9] (Figure 2), a predictive visual code inspired by predictive coding and video codecs: instead of re-encoding every frame, it transmits only what prediction cannot explain. It spends full ViT tokens only on reference frames and encodes the predictable frames in between as compact P-tokens built from motion and residuals, so the model reads an interleaved stream of reference tokens and P-tokens rather than a sequence of full RGB frames. A predictive-cost reset opens a new reference frame whenever prediction becomes costly, placing reference frames exactly where the scene changes enough to need them. This matches our setting directly: a present, always-watching model must consume an unbounded video stream, where a per-frame interface spends full ViT tokens on every frame, so cost and latency grow quickly with the length of the stream. AdaCodec spends only about 16 tokens on each predictable frame and full ViT tokens only at scene changes, so the budget grows far more slowly and its heavy part scales with how much the scene changes rather than with frame count.

3.2 Data Construction for VL-Interaction

The applications we build for share a single demand: the assistant must be present and act on its own, at the moment that matters, with no one there to prompt it. A security feed where a flame has just appeared, a livestream where the item a viewer wants is flashing past, AI glasses that should caption or translate what is in front of you the instant it appears, a companion that should speak up only when there is something worth saying. In each case the right moment to act is set by the world, not by the user, and a model that waits to be addressed has already missed it. The data in this section is built to give the model exactly the abilities these settings demand: to watch a visual stream continuously and decide for itself, second by second, whether to speak, stay silent, or hand a hard problem to a heavier background model; to act on what it sees rather than wait to be asked; and to keep track of time, so that it knows not just what is happening but when. Teaching these abilities broadly, rather than for any single use case, is the goal of everything that follows.

Concretely, at each one-second step the model takes one of three actions over the visual stream seen so far: it can stay silent and keep watching, emitting a `</silence>` token; speak, emitting a `</response>` token and a textual reply; or delegate a slower, harder subtask to an asynchronous background model. Delegation follows a fixed, background-agnostic protocol defined over text requests and results, so any external model, agent, or API can serve as the background (§4). We teach these three decisions at one-second granularity, over and over, across as much of real life as we can capture.

Capabilities the data covers. To span this range, the data comprises more than 4M time-aligned streaming clips organized into six families. The first five are: (1) proactive alerting and anomaly detection on live feeds; (2) time-aligned question answering in backward, present, and forward timing; (3) counting and perception over time; (4) live commentary and narration; and (5) multi-turn casual chat in a variety of styles, ranging from everyday conversation over egocentric short video, to extended dialogue across long videos, to companionship-style exchanges. The sixth (6) cuts across all of the others: delegation episodes pair the ongoing scene with hard, off-stream subtasks—video-grounded knowledge questions, STEM problems, and video reasoning problems—that the model should route to the background rather than answer inline. A single per-second action format applies across every family, so each scenario jointly teaches the model when to speak, when to stay silent, and, where appropriate, when to delegate.

Constructing time-aligned data. Per-second supervision is what makes this behavior learnable, and it is costly to obtain at scale. Every example must be right along two axes: content, what the model says when it speaks, and timing, the exact second at which it should speak, stay silent, or delegate. Silence is a first-class label rather than the absence of one, since most steps in any stream should be silent and the model must learn to wait instead of speaking at every step. To get both axes right at scale, we run a multi-stage pipeline with dedicated verifier agents, and we deliberately favor quality over quantity. Each candidate example is

checked at two levels: globally, over all input frames together with the full annotation, and locally, over the frames at the annotated timestamp and the reply tied to them. An example is admitted only if it passes both checks, and is discarded the moment it fails either. Each source is annotated along whichever axis is its bottleneck, and only examples that survive every check enter the corpus, so heterogeneous raw material is funneled into one clean, consistent per-second action format.

We tailor the construction to each family. For offline video-QA, the content of open-source data is already trustworthy, so the work is almost entirely about timing: a larger VLM marks when the evidence relevant to each question appears, and we form three timing types around it. In a backward example the question is asked after the evidence has appeared and the model responds at the moment the question arrives; in a present example the question is asked just as the evidence appears and the model responds at that moment; in a forward example the question is asked before the evidence appears and the model stays silent until the evidence arrives, then responds at that instant. For multi-turn casual chat, the casual nature of the exchange keeps the bar on content low, so we prioritize timing instead. Two VLM agents converse over a continuously playing video, one asking questions grounded in what it currently sees while the other watches the same stream and answers. To place each turn in time, we sample a time point at random and give the annotating model only the three frames around that point, which anchors every response to its timestamp and prevents large timing drift; a verifier agent then screens each exchange for grounding and quality.

For commentary and narration, the commentator's authentic style is itself the value, so rather than synthesize it we collect open-source commentary and broadcast footage and recover the real spoken content with ASR, which preserves the natural cadence of when a person speaks and pauses and yields realistic silence labels. For counting, we run two passes over a counting-oriented source: the first keeps only videos in which objects reappear, and the second lays down the per-second labels. For perception over time, we insert time-conditioned content into ready-made multi-turn dialogues: for instance, requiring in the question that the answer be withheld for a precise interval before it is given, adding an instruction that increments a running count by one every n seconds, or bounding the time window within which a question and its answer may occur; these rules yield exact, checkable timing without further manual annotation.

For alerting and anomaly detection, timing accuracy matters most, since an alert one second late describes a different moment, so we build this data from two sources. From open-source temporal-grounding annotations we convert directly into per-second labels, with a verification pass over the onsets. For web-collected videos, which carry no annotations, the pipeline proposes a candidate trigger window and tightens it through several stages: it filters videos to viable candidates, samples a window, generates and hard-filters a target by type, and applies a window-level verifier; a dense precheck then scans every 1 fps frame from the query time to the candidate trigger and discards the example if the target appeared earlier, so the alarm marks a genuine first onset, after which we localize the trigger to a single frame.

Anatomy of a delegation episode. Delegation is the most intricate behavior in the corpus, and the most distinctive: it is where the model learns to recognize the limits of real-time inference and hand off, mid-stream, to a heavier background brain without ever pausing the world it is watching. Each episode is built end to end, pairing a trigger (a subtask the model should not solve inline) with a written request and a returned result, and we synthesize these episodes three ways: (i) inserting ready-made pure-text hard problems such as STEM questions, where it is the difficulty rather than the modality that should send them to the background; (ii) converting open-source offline video-reasoning problems and using a large model to weave them into annotated multi-turn chat, so a genuinely hard visual question can surface naturally in the middle of a conversation; and (iii) synthesizing full episodes with a multi-role agent pipeline, in which a Planner scripts the episode, a Timestamp / Visual Verifier runs multi-level checks, over the whole clip and at the trigger frame, that the trigger and its timing are grounded in what is actually on screen, a Background agent produces the heavy answer, and a Foreground Rewriter turns it into a natural reply.

At the heart of the episode is the two-loop choreography it teaches. The instant the model decides to delegate, it does not go quiet. It first hands the user a brief holding reply (for example, "let me look into that"), then emits a hidden delegate token and query that the user never sees, dispatching the hard subtask to the background while the visual stream keeps rolling. We then inject a random delay that simulates the

background's variable reasoning time, and only after that delay is the result folded back in, at which point the model produces its formal answer in context. The delay is the whole point: by varying it, we force the model to stay present while a delegation is still pending, continuing to watch the scene, field new turns, and hold its silence when nothing is worth saying, rather than freezing until the background returns. This is the exact seam where the real-time loop with the user and the asynchronous loop with the background are stitched together, and the model learns to run both at once. Across all six families, however different their raw material, every construction is reduced to the same per-second labels of silence, response, and delegation and passes through the same multi-level verification, which is what lets such heterogeneous sources train a single, unified interaction policy.

A transferable recipe, and signs of emergence. Because the construction is defined by this common per-second format rather than by any one domain, the recipe transfers cleanly to new scenarios: adding a capability is as simple as supplying streams from that domain, annotated in the same per-second form and passed through the same verifiers, with no change to architecture or training objective. And the payoff scales. Even a modest amount of time-aligned data is enough for strong interaction behavior to emerge, a sign that the recipe is highly data-efficient and that time-aligned training is nowhere near saturated, with substantial headroom still ahead as the data grows. Most strikingly, capabilities we never explicitly trained for emerge on their own, including guiding a user through a purchase across changing app screens and improvising a full lecture from a slide deck. We read this as strong evidence that the model is acquiring a general watch-and-interact competence rather than memorizing scenario-specific tricks, and that the same recipe can grow the next generation of always-present assistants, from AI glasses that narrate what you see to companions and accessibility aides that serve as eyes for those who cannot. What we have seen so far is only the beginning.

3.3 Training Recipe of VL-Interaction

Continue training. Starting from JoyAI-VL 1.0, a single supervised training stage turns the base into an interaction model; reinforcement learning then follows below. We mix the time-aligned interaction data of §3.2 into a large pool of conventional turn-based data and fine-tune on the combined corpus. This echoes §3.2: eliciting interaction is data-efficient, and time-aligned training is far from saturated.

Training objective. On time-aligned data, silence steps vastly outnumber the steps on which the model speaks, so the supervised targets are dominated by the `</silence>` token. Under a standard SFT loss this imbalance pushes the gradient toward continued silence and dilutes the signal for responding. Following our streaming-native training [31], we therefore weight the assistant tokens by their role. Let A be the supervised assistant-token positions and $C \subseteq A$ the control-token positions, where each control token is either `</silence>` or `</response>`. We assign $w_{\text{silence}}^{\text{first}}$ to the first `</silence>` in a run, $w_{\text{silence}}^{\text{repeated}}$ to a continued `</silence>`, and w_{response} to `</response>`; every other position gets weight 1. We set $w_{\text{silence}}^{\text{first}} = 1$, $w_{\text{silence}}^{\text{repeated}} < 1$ to down-weight silence continuations, and $w_{\text{response}} > 1$ to up-weight response onsets. A delegation needs no separate weight, because it always rides inside a response. The objective is the normalized weighted cross-entropy

$$\mathcal{L}(\theta) = -\frac{1}{|A|} \sum_{j \in A} w_j \, \log p_\theta(y_j \mid y_{<j}). \tag{1}$$

In practice we use $w_{\text{silence}}^{\text{repeated}} = 0.4$ and $w_{\text{response}} = 1.5$. We apply this weighted loss only to the time-aligned data; the conventional turn-based data is trained with the standard SFT loss.

Reinforcement learning. Continue training teaches the three actions, but the finer points of timing are hard to perfect with token-level supervision: speaking at the right moment, holding silence when nothing is worth saying, and deciding when a subtask is better delegated than answered inline. We add a reinforcement-learning stage, run with GRPO, that optimizes the per-second policy directly against stream-level rewards. Because a long stream would otherwise unfold into hundreds of mostly silent turns, we keep rollouts tractable with answer-centered window sampling: for each gold response we build one trajectory that preserves streaming

causality but retains only the turns that matter for timing, compressing the horizon from hundreds of turns to a handful and concentrating credit where the timing decision lives. The reward credits responses that are both correct and emitted within the right window, rewards appropriate silence, and rewards well-judged delegation; it penalizes false alarms (speaking or delegating with no cause), mistimed responses, and degenerate always-respond behavior. Delegation is scored in two parts: whether the model hands off the genuinely hard subtasks rather than easy ones it should answer inline, and whether it uses the returned result well once the background replies, including staying responsive while a delegation is still pending. Response content is additionally scored for quality by an LLM judge against task-specific rubrics, giving a consistent and fine-grained reward where simple rules fall short.

Infrastructure. We run our RL stage on EasyVideoR1 [21], an efficient reinforcement-learning framework for training vision-language models on video. Its reward system is task-aware, with unified routing and modular extension, so rewards for text, image, video, and our streaming interaction tasks are served from a single pipeline.

4 JoyAI-VL-Interaction System

Around the interaction model of §3 we build a complete, deployable system whose single principle is this: the model is the only component that decides when to speak and when to delegate, while everything else (ASR, TTS, memory, the background brain, the visualization UI) is transduction and orchestration placed around it. Every component ships with an out-of-the-box open-source default and can be swapped for whatever fits a deployment, including the model itself, which any VLM trained with our recipe (§3.3) can replace. This "decision in the model, the rest replaceable" split is the system's design thesis, and it buys two properties at once: the system is composable, so anyone can rebuild it for their own domain, and it is deployable today, because the periphery reuses standard infrastructure rather than a bespoke stack.

Compared with TML, which fuses speech into the model alongside vision, we keep the visual trigger, the model's own decision to act on what it sees, as the native, in-model capability, and treat speech as interchangeable I/O. Placing interchangeable I/O outside the model is therefore not a weakness but a deliberate decoupling of the autonomous core from the parts a deployment will want to choose for itself.

The rest of this section builds it up: the two concurrent loops it runs, the background bridge that joins them, and how that bridge closes a loop from seeing to acting (§4.1); the pluggable transduction, visualization, and user-supplied modules around the model (§4.2); the long-horizon memory that keeps it coherent over hours (§4.3); and the vLLM-native serving path and open release that sustain sub-second presence and put the system in anyone's hands (§4.4).

4.1 Two Concurrent Loops: from Seeing to Acting

The interaction model runs two conversations at once: a real-time loop with the user and an asynchronous loop with the background brain, joined by the model's `delegate` action.

Ingestion. The real-time loop begins at ingestion. A browser client pushes its camera stream to the server through WebRTC SDP negotiation, or submits an RTSP address that the server pulls; the two media paths are received and decoded server-side by aiortc and PyAV respectively, back into raw frames. A sampling module downsamples the stream at a fixed interval, 1 Hz by default and configurable per scenario, to balance temporal detail against real-time latency. The sampled frames are converted to RGB, JPEG-encoded, and wrapped as Base64 data-URLs, then passed with the active user query into an adaptation layer that assembles an OpenAI-compatible multimodal Chat Completions request, the same request form our vLLM serving path expects (§4.4).

The real-time and asynchronous loops. These frames are fed to the model every second, so it always acts on what is happening now rather than on a delayed batch. Together with optional microphone audio transduced by streaming ASR into an active query, they drive the model: at every second it takes one action,

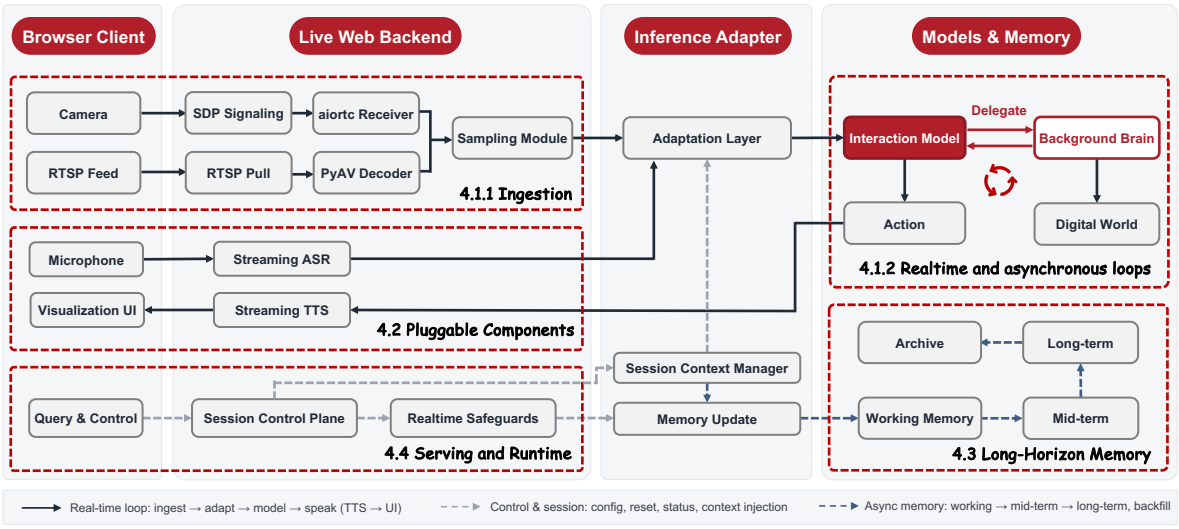

Figure 3 Overview of the JoyAI-VL-Interaction System.

to speak, stay silent, or delegate; when it speaks, streaming TTS renders the reply and the visualization UI reflects its state. When it delegates, control passes to the asynchronous loop, in which a background model runs against the background brain to handle slower, more demanding tasks.

The background bridge. The edge that `delegate` points to is a background bridge, and the brain behind it defaults to the user's own large-model API while remaining free to be any agent (such as Hermes Agent [16] and OpenClaw [18]) the user brings. The scaffold supports this natively and defines a fixed, background-agnostic protocol over text requests and results (§3.2). The bridge exposes a background-agnostic text contract: the foreground emits a tagged query, which the bridge normalizes into a task-id, delegated question, foreground note, and bounded frame snapshot. It runs an isolated model or agent asynchronously under a fixed timeout; expiry is reported as an error event and in-flight work is cancelled on session cleanup. Results return as started/ready/error events, with full artifacts kept off-context and only a bounded digest woven back into the interaction model. Because the background can be an agent, the bridge also closes a loop from seeing to acting: the interaction model observes the physical world and, when warranted, delegates a task the background agent carries out in the digital world. This is the watch-and-do premise of §1 made operational: see, decide, act, in one system.

4.2 Pluggable Components: Transduction, Visualization, and Your Own Modules

ASR and TTS. For speech, we ship a ready-to-run server built on open source ASR [6] and TTS models [5], and users can swap in their own model or API to match their language and preferred voice. These modules only convert modalities; they take no part in the interaction decision. Because speech has duration while generation does not, we predefine a simple rule to keep the two in step: while the previous utterance is still being spoken, the model's next sentence is surfaced as text only and is not synthesized, so audio never piles up behind generation. For high-frequency settings such as live commentary, step-by-step guidance, and real-time subtitle translation, where output is near-continuous, we recommend running with TTS off, and we leave a fuller treatment of this regime to future work.

Visualization UI. A built-in web interface, adapted from NVIDIA's open-source `live-vlm-webui`[1], makes the running system visible and controllable.A left-hand configuration panel selects the video source—either a webcam or an RTSP livestream—and sets the sampling cadence, namely the number of frames per inference and the interval between inferences; it also offers a response mode that shifts the model's style from a default balance to more talkative or more aloof. The center pane shows the live stream the model is watching. The

[1] https://github.com/nvidia-ai-iot/live-vlm-webui

right-hand pane is the conversation: it holds the running exchange with the model, including earlier turns, reports per-response and average latency in real time, and accepts either typed messages or spoken input through the microphone (ASR); when the model speaks, its reply is played back automatically through TTS. The interface doubles as a debugging and trust surface for sensitive deployments, and it can be replaced with a custom front end.

Bring your own modules. Beyond these defaults, the system is open to user-supplied modules that enrich what the assistant perceives or remembers. As one example, a face-recognition module can let the assistant remember who you are and recognize when a particular friend or family member walks into the scene, opening up more personal and engaging interactions. There is a larger reason we open all of this. Turn-based interaction is a long-settled range, refined over years, with a vast ecosystem of systems and optimizations built up around it; VL interaction models, by contrast, have barely begun. By releasing the full stack, we hope to open a first breach for interaction models among those established peaks, and we invite the community to explore with us what this still-young way of working with AI can become.

4.3 Long-Horizon Memory

Long-horizon presence needs a memory whose footprint does not grow without bound as a stream runs for hours, and whose contents form a shared context read by both the streaming model and the background brain. We meet this with a pluggable hierarchical memory built on our prior work [31], organized into tiers at increasing compression. We organize the context into a three-tier hierarchy that partitions the stream into: (i) a *short-term memory* retaining the most recent T_s seconds as raw vision tokens; (ii) a *mid-term memory* holding up to M textual summaries of past short-term chunks, covering $T_m = MT_s$ seconds at moderate compression; and (iii) a *long-term memory* storing up to L aggressively compressed blocks, each consolidated from M consecutive mid-term summaries and thus spanning $T_l = LMT_s$ seconds. Alongside these visual tiers, a dialogue memory keeps past queries and answers coherent across time. The consolidation steps run asynchronously, ahead of each boundary, so they hide behind mainline inference and never stall the real-time loop, and together the tiers reach up to roughly two hours of context. Because each tier above the buffer is stored as text, the memory also forms a stable per-chunk prefix that the serving path of §4.4 can cache and reuse; it is designed for exactly this, and it remains fully pluggable, swappable for any external memory a deployment prefers.

4.4 Serving and Runtime

vLLM-native serving. Hours of continuous video defeat the obvious serving strategies. Recomputing attention over the full history saturates the context window within a few hundred seconds; a sliding window bounds the context but breaks prefix reuse, since successive steps no longer share a common prefix, so an engine's prefix cache buys nothing and latency plateaus above the real-time budget; token pruning has the same problem. Most streaming-video methods inherit one of these and, built on plain Transformers, stay incompatible with efficient engines such as vLLM [10] and SGLang [36], which is much of why they remain in the lab. We instead designed the memory system around prefix reuse so that vLLM serves it directly: the text memory of §4.3 is prefilled once per chunk into a KV cache, and at every subsequent step only the newly observed frames and the previous reply are computed, while the memory and earlier in-chunk turns are reused without recomputation. AdaCodec (§3.1) further shrinks each predictable frame to about sixteen tokens, keeping per-step work small. Building the memory this way is precisely what lowers the bar to deployment: the system sustains over two hours of continuous video at sub-second end-to-end latency on standard vLLM.

Stateful sessions. The runtime is organized around server-side sessions. Control messages, configuration updates, session resets, and state synchronization, travel as JSON over WebSocket or HTTP and carry a `session_id`, while the server keeps each session's video context, question-and-answer trajectory, and memory state. When a request is assembled, this maintained context is injected automatically into the Chat Completions request, so a single round of client-server communication carries only the instantaneous observation, and all long-horizon context management lives at the backend. This is also what makes the prefix reuse above possible: the injected context is precisely the stable per-chunk prefix the engine caches and reuses.

Real-time robustness. To hold latency steady under load, the runtime places timeouts, reconnection, and cancellation across connection setup, the WebSocket channel, RTSP signaling, and the auxiliary audio and video channels, and a session-level lock keeps requests from piling up; when it must, it drops stale frames or backfills late results into the correct interaction history by inference sequence number. The adaptation layer routes each request by its model identifier to the corresponding VLM service, runs the mid-term and long-term memory consolidation of §4.3 in parallel, and returns a standard JSON response.

Everything is open. The model recipe, the background bridge, ASR/TTS, memory, orchestration, the visualization UI, and the vLLM serving path can all be replaced or extended in the repository. Releasing the model recipe and the complete system together is the whole point: from a single repository, anyone can stand up their own present, self-directed real-world assistant.

5 Experiments

5.1 Benchmark

We evaluate JoyAI-VL-Interaction not on offline video-understanding benchmarks but against the real, deployed products people would actually reach for, in the live, event-driven setting this paper is about. Concretely, we compare it head-to-head with the in-app video-call assistants of Doubao and Gemini, today's most mature and recognizable real-time multimodal products (§2.1). We choose six everyday scenarios that fall squarely in the advantage zone of the interaction-model paradigm: settings that demand being present, acting unprompted at the right moment, and tracking events over time, precisely what a turn-based product, waiting to be addressed, is structurally unequipped to deliver however fast it answers once asked.

Scenarios. The six scenarios are: (1) *monitoring and alerting*, flagging an event the instant it occurs; (2) *real-time counting* of objects or events over time; (3) *real-time translation* of on-screen content; (4) *time awareness*, acting on its own sense of elapsed time, for instance speaking once every few seconds on request or timing how long an activity lasts; (5) *live commentary and guidance*, narrating or walking the user through an unfolding scene at the right moments; and (6) *long-horizon memory*, answering about something seen far earlier in the stream. Underneath the six lie the three capabilities that define an interaction model: real-time operation, proactive response driven by what it sees, and long-horizon memory. The six are thus not an arbitrary set but a sampling of these axes, and the same capabilities reach far beyond this benchmark, into AI glasses, assistance for the blind, security monitoring, home robots, home and elder care, vision-grounded chat and companionship apps, live sports commentary, and delegation to background agents, among others. All are settings where being present and acting at the right moment is the whole task, the dimension on which turn-based products structurally fall short, so the six probe the paradigm gap itself rather than any single product feature. The six scenarios comprise 58 cases in total: 10 each for monitoring, counting, translation, and time awareness, and 9 each for commentary and memory. The cases are drawn mostly from public footage on the web, with a few recorded by us.

Baselines. Our baselines are the video-call features of the Doubao and Gemini apps, the products we single out in §2.1 as turn-based at the core, each evaluated as it is actually deployed to end users. We use the app versions current in late May and early June 2026, and drive each product with the same input under matched conditions. We run JoyAI-VL-Interaction through our own system (§4) and compare it against each baseline separately, pairwise. The JoyAI-VL-Interaction system uses a three-tier memory with $T_s = 100\,\text{s}$, $M = 5$, and $L = 15$ (§4.3). At evaluation time, our JoyAI-VL-Interaction system simulates offline videos as a live streaming source over RTSP, served by MediaMTX. Since Doubao and Gemini expose no system-level API, we drive them through their apps: we play the same video to each and pose identical questions at matched timestamps.

Protocol and metric. For every case, human raters score each system on two equally important axes: *quality*, whether the response is correct, relevant, and well-formed; and *timing*, whether it arrives at the right moment, neither premature nor late, and whether the system stays silent when nothing is worth saying. Each axis is

rated on a three-level scale, good, fair, or poor, and a system's score for the case is the equal-weight average of its two axis ratings. The timing axis is the crux of the event-driven setting and the one turn-based products structurally struggle with (§1). To reduce bias, system identities are hidden from raters and the presentation order is randomized. Five raters carry out the ratings, all educated to at least the university level and working as researchers in LLMs, and inter-rater agreement is high. For each case we then compare the two systems' weighted-average scores, which gives a win, tie, or loss for JoyAI-VL-Interaction. We report the per-scenario and overall *win rate*, the fraction of comparisons it wins, against Doubao and against Gemini respectively, alongside the tie and loss rates; a tie counts as neither a win nor a loss.

Scenario	JoyAI-VL-Interaction	Tie	Doubao
Monitoring and alerting	**100.0%**	0.0%	0.0%
Real-time counting	**70.0%**	30.0%	0.0%
Real-time translation	**80.0%**	20.0%	0.0%
Time awareness	**80.0%**	10.0%	10.0%
Live commentary and guidance	**55.6%**	22.2%	22.2%
Long-horizon memory	**77.8%**	22.2%	0.0%
Overall	**77.6%**	**17.2%**	**5.2%**

Table 1 Head-to-head human evaluation, JoyAI-VL-Interaction versus Doubao's in-app video-call assistant.

Scenario	JoyAI-VL-Interaction	Tie	Gemini
Monitoring and alerting	**100.0%**	0.0%	0.0%
Real-time counting	**100.0%**	0.0%	0.0%
Real-time translation	**100.0%**	0.0%	0.0%
Time awareness	**50.0%**	40.0%	10.0%
Live commentary and guidance	**100.0%**	0.0%	0.0%
Long-horizon memory	**77.8%**	22.2%	0.0%
Overall	**87.9%**	**10.3%**	**1.7%**

Table 2 Head-to-head human evaluation, JoyAI-VL-Interaction versus Gemini's in-app video-call assistant.

5.2 Results and Case Study

Experimental Results. Across both comparisons JoyAI-VL-Interaction is the preferred system by a wide margin, and in every scenario it wins more comparisons than it loses. Against Doubao it is preferred in 77.6% of cases, ties 17.2%, and loses only 5.2%; against Gemini the margin is wider still, at 87.9% wins, 10.3% ties, and just 1.7% losses. The two baselines, in other words, almost never come out ahead overall.

Our model's clearest advantage falls exactly on the vision-driven, time-critical scenarios that define the paradigm. In monitoring and alerting, the purest test of catching an event the instant it occurs, it wins every single comparison against both baselines (100%). Real-time translation and fast counting are similarly lopsided: it sweeps both at 100% against Gemini, and wins 80% and 70% against Doubao with no losses at all. These are precisely the settings where being present and acting at the right moment is the whole task, and they are where our margins are largest, which is the central claim of the benchmark borne out in the numbers. Long-horizon memory is also strong, at 77.8% against each. Part of this margin reflects a structural limit of the baselines rather than a single weak answer: in video-call mode Doubao automatically hangs up after about five minutes with no voice input, and Gemini at around two minutes fifteen seconds, so in around

half of the memory cases the question we ask falls past these cutoffs and neither baseline is even present to respond, scoring nothing. However, on the shorter cases that stay within their session limits, our model still wins or ties. We credit it to our long-horizon memory design: a mid-term and a long-term step that continuously compress, merge, and organize what the model has seen, so that information from far earlier in the stream is still at hand when a question finally arrives.

Where the baselines do gain ground, they do so for different reasons, and never on timing. Doubao's main foothold is live commentary and guidance, where it wins 22.2% of cases and ties another 22.2% against our 55.6%. Its edge here is one of quality rather than timing: a larger model scale gives it broader knowledge, richer style, and more varied phrasing, which lands it in the comfort zone of some narration tasks. Its timing, in fact, is a weakness. On commentary, however, its timing works against it: the responses are temporally erratic, arriving too frequently in some passages and too sparsely in others, because its periodic external trigger has no means of judging when a remark is genuinely warranted. This corroborates our central design principle: deciding when to respond must be a native capability of the model, learned and judged internally, rather than a behavior supplied by an external trigger. Doubao therefore prevails on these cases only when its quality advantage is large enough to offset its weaker timing under the equal-weight scoring. Our own losses, in turn, are confined to quality: the model occasionally hallucinates during commentary, a limitation we attribute primarily to its parameter scale and regard as a principal direction for future work. Gemini's single foothold is time awareness, the closest-contested axis against it, where it ties 40% of cases and wins 10%, holding our win rate to 50%, while on every other scenario it fails to win a single comparison. The reason is specific to this scenario: some time-awareness cases are user-triggered, with the question asked only after the relevant moment has passed, so the real-time demand is low. On these ask-after-the-fact questions, Gemini's strong underlying model answers well on quality alone. In both cases the baselines close the gap only where timing pressure is lowest, winning on raw answer quality or on questions that no longer require a split-second response, and they fall furthest behind exactly where a reply must land at the right moment.

Case Study. To make the quantitative gap concrete, we walk through six representative cases; the full side-by-side video comparisons are available in our blog.[2] Each contrasts JoyAI-VL-Interaction with Doubao's and Gemini's in-app video-call assistants on identical input.

In the *Fall Detection Alert* case (02 Monitoring and alerting in the blog), a person collapses on camera. JoyAI-VL-Interaction raises the alert at the instant of the fall, whereas Doubao reacts four to five seconds later, and Gemini explicitly concedes that it is unable to monitor the scene at all. The lag is not incidental: it is the polling interval of an external trigger surfacing as latency on an event that allows no delay. The *Real-time Dart Throw Counting* case (05 Real-time counting in the blog) tests sustained, event-locked counting. JoyAI-VL-Interaction increments its count exactly as each dart strikes the board, registering all six throws on time, while Doubao counts only two and with noticeable delay, and Gemini offers a single "let me check" before falling silent. The *Street Interview Translation* case (01 Real-time translation in the blog) probes continuity, with the task being to translate the interview's on-screen subtitles in real time rather than its audio. JoyAI-VL-Interaction translates the speech continuously and stays accurate even when Chinese pinyin appears in the on-screen subtitles, whereas both Doubao and Gemini translate only what was visible at the moment the request was issued and then stop, treating an ongoing task as a single turn. The *Timed Cooking Scene* case (06 Time awareness in the blog) asks the system to signal once a twenty-second interval has elapsed, a direct probe of an internal sense of time. JoyAI-VL-Interaction is off by only one to two seconds; Doubao fails to signal at the mark altogether, and Gemini does so only at around forty seconds, roughly double the target. The *Pet Livestream Commentary* case (04 Live commentary in the blog) plays a continuous stream of short clips of different pets, requiring the model to keep pace with the changing scene and to recognize and narrate each animal's emotional state. JoyAI-VL-Interaction narrates each pet as it appears, whereas Doubao cannot keep up, describing only three of thirteen and inaccurately at that, and Gemini comments just once, and only when prompted. The *Phone App Delegation* case (09 Agent delegation in the blog) illustrates a capability the baselines do not offer at all: when a phone's app interface appears on screen, the model is asked to produce HTML that renders a similar screen. JoyAI-VL-Interaction recognizes that this exceeds real-time inference, hands the task to the background model, and returns working HTML reproducing the observed

[2] https://joyai-vl-video-future-academy-jd.github.io/JoyAI-VL-Interaction

interface, all without leaving the live session.

Emergent capabilities. Two further cases are more striking still, in that they exercise capabilities the model was never trained for. In the *Shopping App Guidance* case (03 App guidance in the blog), the user pursues a shopping goal while swiping through a phone's screens. JoyAI-VL-Interaction responds promptly and narrates in step with each swipe, guiding the user to the intended item, whereas neither Doubao nor Gemini reacts proactively or in real time, describing only a few of the screens first shown. Notably, our training data contains no app-interface video of any kind: the ability to follow a changing interface is one the model acquired on its own, generalizing to a domain it never saw. In the *Travel Scene Commentary* case (04 Live commentary in the blog), the system is asked to narrate once every four seconds. JoyAI-VL-Interaction holds to this cadence throughout and keeps the commentary substantively strong; Doubao narrates only the opening scene for some twenty seconds and ignores the four-second instruction entirely, while Gemini, though asked in Chinese, replies in English and comments only once. The task combines two abilities, timed action and live commentary, that never co-occur in our training data: the model composes them at inference time, an emergent crossing of capabilities rather than a pattern it was shown.

Across these cases the same factor separates JoyAI-VL-Interaction from systems many tens or even hundreds of times its size: timing. By making the decision of when to respond a native, learned capability of the model rather than a behavior imposed by an external trigger, JoyAI-VL-Interaction acts at the moment each event demands, where larger but turn-based or polling-based systems arrive late, stop after a single turn, or never engage. Moreover, its compact size imposes no ceiling on what it can ultimately handle: when a problem genuinely calls for a more powerful model, JoyAI-VL-Interaction does not attempt to answer it inline but delegates it, dispatching the hard subtask to a background model through an asynchronous request and folding the result back in once it returns, all while remaining present with the user in real time. A compact 8B model thus pairs real-time, well-timed presence with on-demand access to heavier reasoning, which we regard as the decisive reason it can outperform far larger models and mature products on the dimension that matters most in the streaming setting.

5.3 Limitations

A fair reading of our comparison has to start with scale. The two video-call assistants we evaluate against are powered by far larger models, Seed 2.0 behind Doubao and Gemini-3.1-flash-live behind Gemini, and both are mature products tuned over time against real users and use cases. JoyAI-VL-Interaction, by comparison, is a compact 8B-scale model. We therefore expect, and do not claim otherwise, that on general turn-based ability these products are stronger than ours: broader world knowledge, a more polished one-on-one chat experience, and greater robustness on complex or rarely seen inputs.

Our claim is narrower, and we think more telling. In the six event-driven scenarios this paper targets, the ones that turn on real-time presence, proactive interaction, and a sense of time, and that already carry rich deployment value, a far smaller model holds a natural advantage simply by being an interaction model rather than a turn-based one. That a compact, open model can outperform far larger and heavily optimized products precisely in this regime is, to us, the point: interactivity is worth scaling as a capability in its own right, and it begins to pay off well before enormous scale.

Both the data and the evaluation behind this release are, by deliberate choice, still at an early stage. On the data side, we have not yet tuned the mixture or scaled it to the volume we ultimately intend, and a further round of cleaning remains; one consequence is already visible in our results, where the comparatively sparse commentary data, together with the model's compact scale, is the main cause of the occasional hallucination during live narration (§5). The evaluation is similarly preliminary, six scenarios and 58 human-rated cases against two products, rather than the larger, more fine-grained study we ultimately want. We release at this stage regardless, and the reason is the same for both: even with this early data and this limited evaluation, the model already exhibits interaction capabilities we never explicitly trained for, and watching them emerge convinced us, more than any single number could, that interactivity is a direction worth scaling in its own right. That conviction is exactly why we would rather put the model, the recipe, and the system into the community's hands now than hold them back for a polished data recipe and a larger benchmark: we are eager

to scale this direction together rather than alone. A subsequent version will tune the data mixture, scale the corpus, and report a broader and more systematic evaluation; in the interim, it is the data-construction methodology and the approach itself, rather than this particular corpus or benchmark, that we expect others can most readily adapt and extend. These limitations also chart the road ahead: closing the gap in general ability and robustness, widening the set of scenarios, and pushing further on what an interaction model can do.

6 Conclusion

We set out to move large models past the turn-based paradigm, in which a model opens its eyes only when it is addressed, toward genuine streaming interaction, in which the model is present in the world and decides for itself when to act. To that end we release, to our knowledge, the first open vision-driven interaction model, together with everything needed to build on it: the time-aligned data, the training recipe, the 8B model, and a complete, deployable system.

At the center is a single conviction, that interactivity should scale as a capability of the model itself rather than be simulated by a harness around it. Scaling interactivity means scaling three things at once: the model's freedom to speak on its own when a moment is worth a word, its ability to interact in real time, and its sense of elapsed time. As these grow, a model behaves less like a tool that waits to be queried and more like a participant that is simply present, watching what is happening and responding as a person would. We see this as a direction of scaling in its own right, alongside the scaling of intelligence, and one that brings models closer to being genuinely present in human settings.

Across the streaming scenarios we study, we already see early signs that an interaction model carries natural advantages for real deployment, from monitoring and live narration to step-by-step guidance and companionship. The moment we are reaching for is an everyday one: you come home worn out after a long day, and before you have said anything, a quiet voice notices and offers, "I can see you're tired; today must have been hard on you." Presence like that, given unasked, is what an interaction model makes possible and a turn-based one, waiting to be addressed, never can. Much remains open in both the model and the system, and we have released the whole stack openly to lower the barrier and accelerate this shift. Turn-based models, products, and optimizations have been built up over many years, while vision-driven interaction models are only beginning; if this release opens even a first foothold for them among those long-settled peaks, and draws others into building it with us, it will have done its job. We invite the community to explore, with us, what a model that is truly present in the world can become.

References

[1] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. https://arxiv.org/abs/2511.21631.

[2] ByteDance Seed. Seed2.0 Model Card: Towards Intelligence Frontier for Real-World Complexity. Available at ByteDance Seed Model Cards, 2026.

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

[4] Chung-Ming Chien, Manu Orsini, Eugene Kharitonov, Neil Zeghidour, Karen Livescu, and Alexandre Défossez. Moshirag: Asynchronous knowledge retrieval for full-duplex speech language models, 2026. https://arxiv.org/abs/2604.12928.

[5] Zhihao Du, Changfeng Gao, Yuxuan Wang, Fan Yu, Tianyu Zhao, Hao Wang, Xiang Lv, Hui Wang, Xian Shi, Keyu An, et al. Cosyvoice 3: Towards in-the-wild speech generation via scaling-up and post-training. arXiv preprint arXiv:2505.17589, 2025.

[6] Zhifu Gao et al. Funasr: A fundamental end-to-end speech recognition toolkit. In INTERSPEECH, 2023.

[7] Google. Gemini 3.1 Flash Live: Making Audio AI More Natural and Reliable. Google Blog (The Keyword), March 2026. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-flash-live/. Posted 2026-03-26. Accessed 2026-06-06.

[8] 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. arXiv preprint arXiv:2603.12262, 2026.

[9] Haowen Hou, Zhen Huang, Zheming Liang, Qingyi Si, Chenglin Li, Shuai Dong, Kele Shao, Ruilin Li, Dianyi Wang, Nan Duan, and Jiaqi Wang. Adacodec: A predictive visual code for video mllms, 2026. https://arxiv.org/abs/2606.02569.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. https://arxiv.org/abs/2309.06180.

[11] Thinking Machines Lab. Interaction models: A scalable approach to human-ai collaboration. Thinking Machines Lab: Connectionism, May 2026. doi: 10.64434/tml.20260511. https://thinkingmachines.ai/blog/interaction-models/.

[12] Yifei Li, Junbo Niu, 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?, 2025. https://arxiv.org/abs/2501.05510.

[13] 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, 2024. https://arxiv.org/abs/2411.03628.

[14] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026. https://arxiv.org/abs/2601.05242.

[15] Kevin Lu and Thinking Machines Lab. On-policy distillation. Thinking Machines Lab: Connectionism, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

[16] Nous Research. Hermes agent: A self-improving open-source ai agent. https://github.com/NousResearch/hermes-agent, 2026. GitHub repository.

[17] OpenAI. Advancing voice intelligence with new models in the API (GPT-Realtime-2). OpenAI Blog, May 2026. Announced 2026-05-07. https://openai.com/index/advancing-voice-intelligence/. Accessed 2026-06-02.

[18] OpenClaw Foundation. Openclaw: An open-source autonomous ai assistant. https://github.com/openclaw/openclaw, 2026. GitHub repository.

[19] 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. arXiv preprint arXiv:2501.03218, 2025.

[20] Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. Advances in Neural Information Processing Systems, 37: 119336–119360, 2025.

[21] Chuanyu Qin, Chenxu Yang, Qingyi Si, Naibin Gu, Dingyu Yao, Zheng Lin, Peng Fu, Nan Duan, and Jiaqi Wang. Easyvideor1: Easier rl for video understanding, 2026. https://arxiv.org/abs/2604.16893.

[22] Qwen Team. Qwen3.5-omni technical report, 2026. https://arxiv.org/abs/2604.15804.

[23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. https://arxiv.org/abs/2402.03300.

[24] Volcano Engine (ByteDance). Real-time conversational AI: Video and image understanding. Volcano Engine Documentation, 2026. https://www.volcengine.com/docs/6348/1408245. Accessed 2026-06-02.

[25] Yiyu Wang, Xuyang Liu, Xiyan Gui, Xinying Lin, Boxue Yang, Chenfei Liao, Tailai Chen, and Linfeng Zhang. Accelerating streaming video large language models via hierarchical token compression, 2026. https://arxiv.org/abs/2512.00891.

[26] 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, 2024. https://arxiv.org/abs/2411.17991.

[27] Jiaer Xia, Peixian Chen, Mengdan Zhang, Xing Sun, and Kaiyang Zhou. Streaming video instruction tuning, 2025. https://arxiv.org/abs/2512.21334.

[28] Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. Streamingvlm: Real-time understanding for infinite video streams, 2025. https://arxiv.org/abs/2510.09608.

[29] Weicai Yan, Yuhong Dai, Qi Ran, Haodong Li, Wang Lin, Hao Liao, Xing Xie, Tao Jin, and Jianxun Lian. Proact-vl: A proactive videollm for real-time ai companions, 2026. https://arxiv.org/abs/2603.03447.

[30] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. https://arxiv.org/abs/2505.09388.

[31] 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, 2026. https://arxiv.org/abs/2606.08615.

[32] 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, pages 10807–10816, 2025.

[33] ydyhello. Awesome-VLM-Streaming-Video: A Curated Collection of Papers and Open-Source Code on Vision-Language Models for Streaming Video. GitHub repository, 2026. https://github.com/ydyhello/Awesome-VLM-Streaming-Video. Accessed 2026-06-06.

[34] 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 The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. https://openreview.net/forum?id=9loSPaBwGO.

[35] Haowei Zhang, Shudong Yang, Jinlan Fu, See-Kiong Ng, and Xipeng Qiu. Hermes: Kv cache as hierarchical memory for efficient streaming video understanding, 2026. https://arxiv.org/abs/2601.14724.

[36] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. https://arxiv.org/abs/2312.07104.

7 Appendix

7.1 Training Data Example

```
{
  "role": "user",
  "content": "Alert me whenever a physical confrontation happens. \n<0.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<1.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<2.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<3.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<4.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> A physical confrontation is happening."
},
{
  "role": "user",
  "content": "<5.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> A physical confrontation is happening."
},
{
  "role": "user",
  "content": "<6.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> A physical confrontation is happening."
},
{
  "role": "user",
  "content": "<7.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> A physical confrontation is happening."
},
{
  "role": "user",
  "content": "<8.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> A physical confrontation is happening."
},
{
  "role": "user",
  "content": "<9.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> A physical confrontation is happening."
},
{
  "role": "user",
  "content": "<10.0 seconds>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
...
```

Listing 1 Example of monitoring and alerting.

```
...
{
  "role": "user",
  "content": "Is the person cleaning the desk?\n<7.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> Yep, definitely cleaning the desk! Getting all the dust off that pink surface."
},
{
  "role": "user",
  "content": "<8.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<9.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<10.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<11.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<12.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "What are they doing with the pillow?\n<13.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> They're picking up the pillow and moving it around, maybe fluffing it or putting it somewhere."
},
{
  "role": "user",
  "content": "<14.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<15.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<16.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<17.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "Why is the person holding keys now?\n<18.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> Hmm, they seem to be fiddling with the keys on the desk. Maybe checking or organizing them?"
},
...
```

Listing 2 Example of multi-turn casual chat.

```
...
{
  "role": "user",
  "content": "<29.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "What are those two guys doing near the boat?\n<30.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> Looks like they're walking alongside it, probably supervising or guiding as it gets lifted."
},
{
  "role": "user",
  "content": "<31.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<32.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<33.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<34.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<35.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<36.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "<37.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
},
{
  "role": "user",
  "content": "Why does this boat hoist use a four-point sling configuration with spreader bars rather than lifting from a single central point, and how does the load distribution
        prevent hull deformation in fiberglass vessels?\n<38.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</response> Let me check the engineering principles behind that hoist design. </delegation> Explain the structural engineering principles of four-point boat lifting
        slings, load distribution, and hull deformation prevention in fiberglass boat hoists."
},
{
  "role": "user",
  "content": "<39.0 seconds>\n<image>\n<image>\n<image>\n<image>"
},
{
  "role": "assistant",
  "content": "</silence>"
...
```

Listing 3 Example of background model delegation.