ONSTEP: LEARNING PROACTIVE AND INTERACTIVE ASSISTANCE IN STREAMING EGOCENTRIC VIDEO

Anonymous authors
Paper under double-blind review

ABSTRACT

Everyday procedural tasks require assistance that is timely, context-aware, proactive, and interactive. Although recent multimodal large language models (MLLMs) and streaming video assistants have advanced video understanding and real-time interaction, they still fall short of providing unified procedural assistance that tracks task progress, proactively intervenes at the right moment, and responds to user queries as the task unfolds. We instantiate this setting in egocentric cooking, a procedurally rich and error-prone domain with large-scale, step-annotated videos capturing both task progress and execution errors. To support learning and evaluation in this setting, we introduce **ProactiveCooking**, a temporally grounded dataset that augments proactive cooking demonstrations with 6,526 human-verified question-answer interactions spanning six complementary interaction types. ProactiveCooking provides both training supervision and a streaming evaluation testbed, covering standard recipe execution (Main set) as well as more challenging non-monotonic execution (Advanced set) with reordered, repeated, and skipped steps. Building on the training data, we introduce **ONSTEP** (**ON**set-gated, **STEP**-anchored Proactive guidance), an end-to-end streaming framework that explicitly models when to begin speaking, how long an utterance should continue, and where the user is in the task procedure. ONSTEP provides step-by-step instructions, visually grounded feedback on completed actions and mistakes, and real-time answers to user questions within a unified streaming loop. Across both the Main and Advanced test sets, ONSTEP consistently improves speaking-time F1 over the strongest proactive baseline from 35.04 to 39.70 and from 32.79 to 37.50, respectively.

1 INTRODUCTION

Much of everyday life is procedural: we follow recipes, assemble furniture, and repair appliances, one step at a time. Effective assistance for such tasks should go beyond static instructions or one-shot answers. Cooking is a canonical example: the user's hands and attention are occupied, the environment changes continuously, and the difference between useful and ineffective guidance is often a matter of *timing*. For example, an assistant that says "lower the heat" after the garlic has burned has failed, because guidance delivered too late is no longer useful.

Recent multimodal large language models (MLLMs) have advanced video understanding and question answering (Bai et al., 2025; Liu et al., 2023), while streaming assistants process incoming rather than prerecorded video (Yan et al., 2026; Chen et al., 2024). Most closely, Bhattacharyya et al. (2025) study live step-by-step guidance, where a model issues instructions and feedback at appropriate times. This formulation is one-directional: it does not support questions such as *How much salt should I add?* or *Which pan should I use?* Conversely, turn-based video QA responds to prompts but does not proactively instruct or detect mistakes. Practical assistance requires both capabilities in one streaming loop.

A central obstacle to this goal is the lack of suitable training data. Supervising a proactive and interactive assistant requires examples that are both *temporally grounded*, so that instructions, feedback, and answers are aligned with when they should be delivered, and *semantically grounded*, so that responses are supported by the current visual state or task context. Existing cooking demonstrations

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.

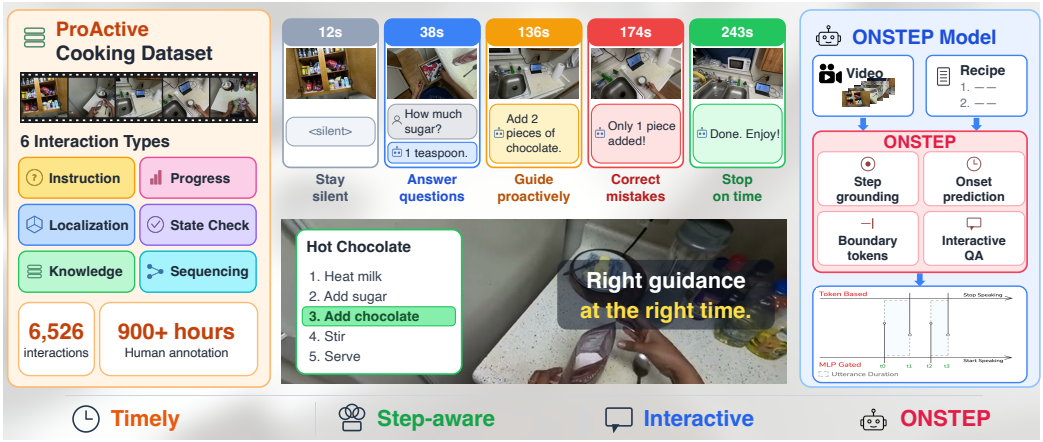


Figure 1: **Overview of ProactiveCooking data and ONSTEP model.** The left section shows our ProactiveCooking , the middle section shows real ONSTEP output, while the right section highlights the streaming model that decides when to speak, what to say, and when to stop.

provide much of the instruction–action–feedback structure, but rarely include the user questions that arise during task execution together with their corresponding answers.

To fill this gap, we introduce **ProactiveCooking** , a temporally grounded dataset and benchmark for proactive and interactive assistance, containing 6,526 human-verified question–answer interactions. Starting from existing cooking demonstrations, we construct ProactiveCooking through a three-stage generation-and-verification pipeline, as in the left part of fig. 1. An agentic LLM first proposes time-anchored question–answer turns spanning six complementary interaction types, covering instruction details, visual grounding, domain knowledge, procedural progress, mid-step state, and other task-relevant visual information. Human reviewers then filter invalid or irrelevant proposals, after which annotators verify both the content and timing of each retained pair against the video.

Using this dataset, we introduce **ONSTEP** (**ON**set-gated, **STEP**-anchored proactive guidance), a streaming model fine-tuned from Qwen3-VL (Bai et al., 2025) that guides users through goal-directed procedures while deciding when to speak. It provides timely instructions and action feedback while answering questions from the observed video and task context. To achieve this, ONSTEP combines three components. First, an *onset head* is supervised only at the beginning of each assistant utterance, explicitly modeling when to start speaking. Second, boundary tokens, <|cont|> and <|done|>, determine whether an utterance continues into the next chunk or terminates. Third, a step-anchored contrastive objective aligns the streaming representation with an order-aware representation of the current procedural step. Together, these components model *when to start*, *when to continue or stop*, and *the current procedural step* within a unified streaming framework. The central and right parts of fig. 1 illustrate our model and its streaming behavior.

To comprehensively evaluate the effectiveness of ONSTEP , we assess not only *what* the assistant says, but also *when* it speaks. F1, TimeDiff, and pAUC assess the temporal alignment of assistant responses, while LLM-based metrics evaluate answer quality, online utility, and task quality. Most notably, compared with the strongest baseline, ONSTEP improves speaking-time F1 from 35.04 to 39.70 on the Main testing set and from 32.79 to 37.50 on the Advanced testing set with non-monotonic execution. These timing gains are accompanied by improvements in most delivery-quality metrics while maintaining competitive answer quality.

In summary, our contributions are:

- **A temporally anchored, grounded interactive cooking dataset.** ProactiveCooking is a benchmark made of six types of question-answer turns using an agentic generation pipeline followed by two stages of human verification.
- **A unified proactive and interactive guidance system.** We introduce ONSTEP , which uses onset-prediction with step-anchored loss to provide real-time guidance.
- **Consistent gains in timing and assistance quality.** Evaluation shows that ONSTEP outperforms baselines on both guidance timing and response quality.

2 Related Work

Egocentric and Procedural Activity Datasets. Egocentric datasets capture activities from the actor's viewpoint: EPIC KITCHENS (Damen et al., 2022) focuses on unscripted cooking, Ego4D (Grauman et al., 2022) spans everyday activities, and Ego-Exo4D (Grauman et al., 2023) pairs egocentric and exocentric views across skill levels. Their retrospective annotations describe what happened but not when an assistant should intervene. The Qualcomm Interactive Cooking (QIC) benchmark (Bhattacharyya et al., 2025) extends CaptainCook4D (Peddi et al., 2024) with temporally localized instructions and feedback, but remains unidirectional. ProactiveCooking contains temporally grounded question-answer interactions, as summarized in Table 9.

Video Multimodal Large Language Models. MLLMs have extended visual instruction tuning from images (Alayrac et al., 2022; Liu et al., 2023) to video (Li et al., 2025a; 2023). Both closed-source systems, including GPT-4 and Gemini (Achiam et al., 2023; Comanici et al., 2025), and open-weight models, including Qwen3-VL and LLaVA-OneVision (Bai et al., 2025; Liu et al., 2024a), demonstrate strong temporal reasoning and video-grounded dialogue. However, despite this progress, most Video MLLMs remain optimized for offline, post-hoc QA over sparse frames, limiting their ability to handle streaming video. In contrast, ONSTEP provides proactive and interactive guidance over a continuously arriving video stream.

Streaming and Proactive Video Assistants. Streaming video assistants (Chen et al., 2024; Liu et al., 2024b) differ in both latency and agency. Proactive models decide when to speak but often process video offline or at reduced rates, including VideoLLM-Online (Chen et al., 2024), MMDuet2 (Wang et al., 2026), LiveStar (Yang et al., 2025b), and TimeChat-Online (Yao et al., 2025). Real-time models such as LiveCC (Chen et al., 2025) and StreamingVLM (Xu et al., 2026) keep pace with the stream but provide continuous commentary. StreamMind (Ding et al., 2025), Mage-VL (Yang et al., 2026), JoyAI-VL (Yao et al., 2026), and Proact-VL (Yan et al., 2026) combine real-time processing with learned speak-or-silence decisions. Existing benchmarks (Li et al., 2025b; Yang et al., 2025a) nevertheless emphasize prompted VideoQA. ONSTEP instead targets goal-directed procedural assistance that must decide when to intervene, provide grounded guidance, and answer questions during task execution.

3 ProactiveCooking Dataset

A benchmark for proactive and interactive guidance requires three elements: videos of procedural task execution, temporally grounded guidance, and user interaction. Existing benchmarks typically provide the first two, but rarely capture proactive assistance and user-initiated interaction within the same streaming setting. We therefore extend existing proactive cooking demonstrations with temporally grounded user–assistant interactions, forming ProactiveCooking .

3.1 Data Source and Temporally Grounded Interaction

We build ProactiveCooking on the egocentric cooking videos from CaptainCook4D (Peddi et al., 2024) and the temporally aligned instruction–feedback transcripts from the Qualcomm Interactive Cooking (QIC) benchmark (Bhattacharyya et al., 2025). In QIC, the assistant issues step-by-step instructions and later provides localized feedback that confirms completion or identifies mistakes, providing natural streaming supervision for proactive guidance. However, QIC is entirely one-directional: every utterance comes from the assistant, so models trained on it do not learn how to respond when users ask questions during task execution, such as "How much should I leave at the edge?", "Where is the paper towel?", or "How many steps are left?" Answering such questions requires jointly reasoning over the current instruction, the user's progress through the recipe, and the observed visual state. We therefore augment QIC with interactive user question–answer turns while preserving all original annotations.

3.2 Generation and Verification Pipeline

We organize interactive questions into six types covering instruction-specific values, visually grounded object queries, culinary knowledge, recipe progress, mid-step progress, and other step-

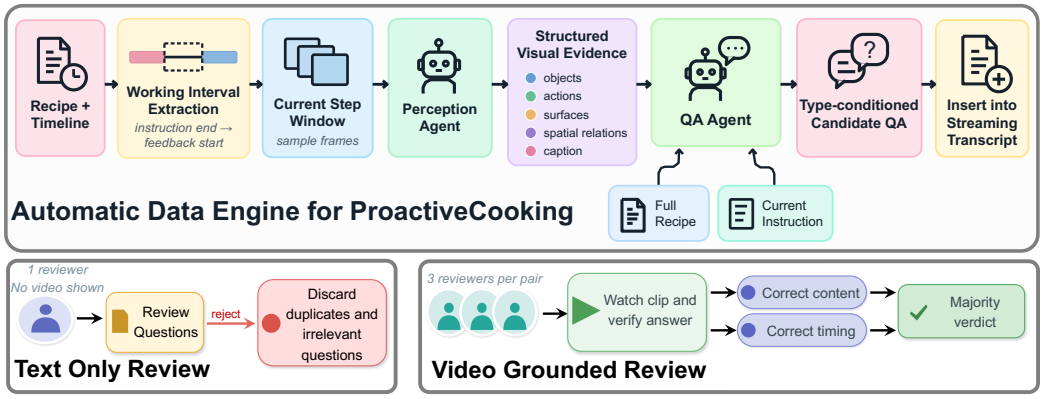


Figure 2: **Data generation pipeline of ProactiveCooking .** Candidate question-answer pairs are generated within temporally defined working intervals, filtered by text-only review, and then verified against the corresponding video by human annotators for both content and timing.

relevant visual information. The complete taxonomy, including the target and answer source for each type, is provided in the appendix (section D.6).

Stage 1: automatic generation. We begin with each recipe and its temporally aligned streaming transcript. From this timeline, we extract a *working interval* for every step, defined as the period between the end of the instruction and the beginning of its closing feedback. This interval represents the time during which the user is executing the instructed action.

For each working interval, we form a current-step window and uniformly sample up to 20 frames from the portion observed by the perception agent. The perception agent converts these frames into structured visual evidence describing visible objects, actions, surfaces, spatial relations, and an overall scene caption. This intermediate representation makes the evidence used for question generation explicit and auditable.

A QA agent then receives the structured visual evidence together with the full recipe and current instruction. It produces type-conditioned candidate question-answer pairs only when the corresponding prerequisites are satisfied. For example, an `instruction_spec` question requires an explicit value in the instruction, visual questions require supporting frame evidence, and `mid_step_state` questions apply only to passive cooking or waiting steps. Most questions are placed shortly after the instruction, whereas mid-step questions may be generated at multiple points to capture changes in task progress. The resulting interactions are inserted into the streaming transcript, with programmatic checks ensuring that they neither overlap existing utterances nor extend beyond the video.

Stage 2: text-only review. Each candidate interaction is screened by a human reviewer who sees the question, proposed answer, and the current instruction, but not the video. The reviewer removes duplicate, irrelevant, incoherent, or otherwise unanswerable questions. This stage filters failures that can be identified from text alone before the more expensive video-grounded review.

Stage 3: video-grounded verification. Each surviving question-answer pair is assigned to three independent annotators. Annotators view a clip spanning the instruction, its execution, and the closing feedback, together with the candidate question and answer. They separately verify the answer content and the question timing. An annotator may accept or correct the answer, reject the interaction, or mark it as uncertain; mistimed questions may also be assigned a corrected timestamp. Final labels are determined by a two-of-three majority, with an additional annotator adjudicating cases that do not reach a majority.

Advanced Planning videos require special handling because corrective feedback can be followed by a verbatim reissue of the same instruction. We treat such repetitions as part of a single step and keep the review clip open until a different instruction begins. This prevents the clip from ending before the complete execution and its associated feedback are visible. Further details on clip construction and data processing are provided in the appendix section D.

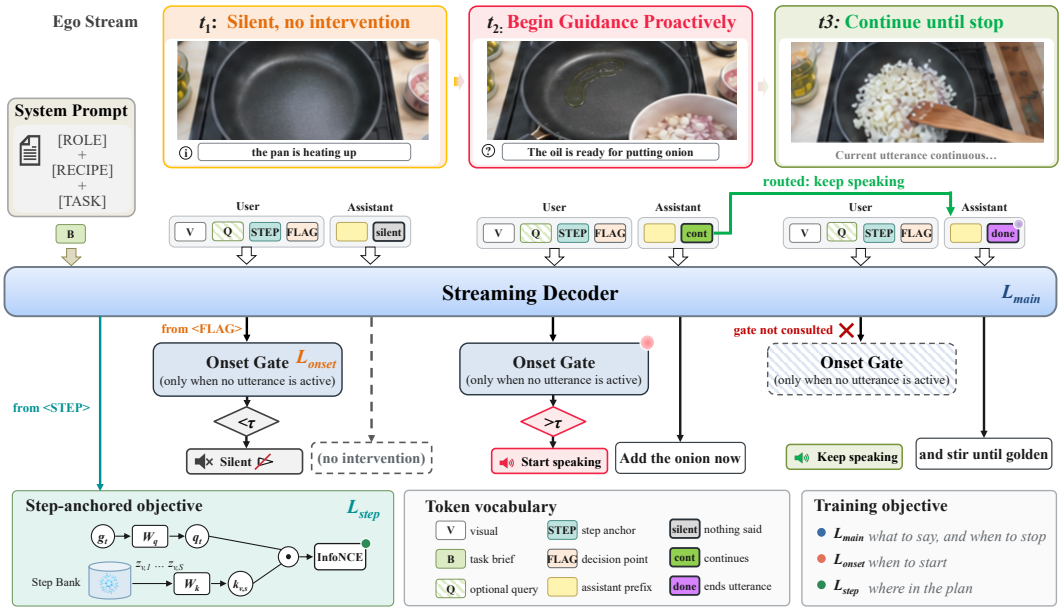


Figure 3: ONSTEP overview. The top section follows three consecutive chunks of an egocentric stream, staying silent, beginning guidance proactively, and continuing an utterance already under way, while the bottom section highlights the onset gate, the step-anchored objective, and the boundary tokens that decide *when to start*, *where in the plan*, and *when to stop*

3.3 Dataset Statistics

For the text-only review, 768 question-answer interactions were rejected from 7,759. Rejected examples are shown in section A.5. For video-grounded verification stage, 6,942 candidate interactions received 20,634 individual judgments from 125 annotators. After verification, ProactiveCooking contains **6,526 question-answer pairs**, comprising 5,510 model-generated answers accepted by annotators and 1,016 answers corrected during review. These interactions cover 679 videos: all 384 Main-set videos and 295 of the 302 Advanced-set videos. They span 6,196 distinct instruction steps, corresponding to an average of 9.6 interactions per video and approximately one interaction per executed step. Per-type statistics, reviewer agreement, timing corrections, and further annotation analyses are reported in the appendix section A.

Agreement. 4,907 items (70.7%) were unanimous, 1,833 (26.4%) resolved by majority with one dissenting reviewer, and 153 (2.2%) reached no majority. Among the 77 annotators with 100 reviews or more, per-rater accept rates span 55% to 93% around a median of 78%, a 1.7× spread between the strictest and the most lenient.

4 Proposed ONSTEP Assistant

Given a first-person video stream and an ordered task plan, the model must decide when to offer guidance, what to say, and how to answer user questions. Building on the chunk-wise interface of Yan et al. (2026), ONSTEP introduces an onset gate for beginning an utterance, boundary tokens for continuing or ending it, and a step-anchored objective for tracking procedural position (fig. 3).

4.1 Streaming Interface

A video is discretised into one-second chunks. Each session is conditioned on a task brief B, the assistant's role together with the ordered procedure it must guide the user through.At time t the model receives (V_t, Q_t), the visual content of the current one-second window, and an optional user

query, and emits an utterance segment U_t while updating a persistent key-value cache:

$$(U_t,\ K_t)\ =\ f_\theta(V_t,\ Q_t,\ K_{t-1},\ B)\,. \tag{1}$$

U_t is written into K_t and conditions step $t+1$, so an utterance longer than one second flows naturally across consecutive chunks A decision token <|FLAG|> terminates every user message; we write $h_t \in \mathbb{R}^d$ for its hidden state at layer ℓ_a, and read all speaking onset decisions off that position.

4.2 Onset detection

The timeline decomposes into three regimes. Let the reference transcript for a video contain M utterances $\mathcal{U} = \{u_1, \ldots, u_M\}$, where u_m occupies the contiguous span $[\alpha_m, \beta_m]$ of seconds. This induces a partition of the timeline:

$$\mathcal{O} = \{\alpha_m\}_{m=1}^{M} \qquad \text{onsets: an utterance begins,} \tag{2}$$

$$\mathcal{C} = \textstyle\bigcup_m \{\alpha_m + 1, \ldots, \beta_m\} \qquad \text{continuations: one is already running,} \tag{3}$$

$$\mathcal{S} = \{1, \ldots, T\} \setminus (\mathcal{O} \cup \mathcal{C}) \qquad \text{silence.}$$

These three regimes pose different questions. At $t \in \mathcal{S}$ the model must detect that the scene now warrants speech, a fully perceptual judgment about the video. At $t \in \mathcal{C}$ the question is, whether the sentence already underway has finished, which is a property of both, the text being produced and of from the current frames.

The distinction is not a corner case. Utterances span several seconds, so continuation dominates: in our data $|\mathcal{C}|/(|\mathcal{O}| + |\mathcal{C}|) = 58.7\%$. A per-second speaking indicator therefore devotes the majority of its positive supervision to the easier of the two questions, and forces a single decision boundary to serve both.

An onset-only target. We give the gate the first question alone. With $\perp$ denoting an ignored position,

$$y_t\ =\ \begin{cases} 1, & t \in \mathcal{O} \quad \text{(begin an utterance)} \\ 0, & t \in \mathcal{S} \quad \text{(remain silent)} \\ \perp, & t \in \mathcal{C} \quad \text{(gate not consulted)} \end{cases} \tag{4}$$

and $\mathcal{M} = \{t : y_t \neq \perp\} = \mathcal{O} \cup \mathcal{S}$ is the supervised set. The gate itself is a linear probe with a sigmoid,

$$p_t\ =\ \sigma\big(w^\top h_t + b\big)\,, \qquad a_t\ =\ \mathbb{I}[p_t \geq \tau]\,, \tag{5}$$

where τ is a decision threshold and $a_t = 1$ triggers generation. Under equation 4 the gate models a single homogeneous event-detection problem: every positive is the leading edge of an utterance. Positives are sparse, 5.3% of seconds in our data, which we handle with class weighting.

4.3 Termination as a generated token

Masking $\mathcal{C}$ leaves utterance length unmodelled, so we make it explicit in the token stream. This separates the decision of *when to begin speaking*, handled by the onset gate, from *how long to continue speaking*, handled by generated boundary tokens. Every assistant turn ends with exactly one **boundary marker**:

$$m_t = \begin{cases} \texttt{<|cont|>}, & t < \beta_{m(t)} \quad \text{(the utterance continues)} \\ \texttt{<|done|>}, & t = \beta_{m(t)} \quad \text{(the utterance ends here)} \end{cases} \tag{6}$$

with $m(t)$ indexing the utterance containing t. The two markers are distinct single tokens added to the vocabulary. Since, m_t is the final token of the assistant turn, so it is supervised by the ordinary language-modelling objective and adds no term to the loss and no parameters to the model. Termination is learned exactly as any other token is. At silent time steps, no assistant content is decoded. Instead, the router deterministically appends a <|silent|> token to the model context, allowing subsequent chunks to condition on the passage of a silent step. These positions are masked from the language-modelling loss: the model observes <|silent|> as context but is never trained to generate it.

4.4 STEP-ANCHORED CONTRASTIVE OBJECTIVE

Procedural guidance depends on the user's position in the task plan. We therefore train the streaming representation to identify both the active instruction and its position in the ordered recipe.

For video v, let $(r_{v,1}, \ldots, r_{v,S_v})$ denote its ordered step descriptions. We construct an order-aware representation for each step from its causally encoded recipe prefix:

$$z_{v,s} = E_{\text{step}}\left(r_{v,1} \oplus \cdots \oplus r_{v,s}\right)_{\text{last}}, \qquad s \in \{1, \ldots, S_v\}, \tag{7}$$

where E_{step} is a frozen causal text encoder. Encoding the prefix rather than the instruction alone distinguishes repeated instructions that occur at different positions in the procedure.

Let g_t be the streaming hidden state of <|STEP|> and c_t the active step at time t, obtained from the aligned transcript. We project and normalize g_t and $z_{v,s}$ into q_t and $k_{v,s}$, respectively, and optimize a within-video InfoNCE objective:

$$\mathcal{L}_{\text{step}} = -\frac{1}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} \log \frac{\exp\left(\gamma q_t^\top k_{v,c_t}\right)}{\sum_{s=1}^{S_v} \exp\left(\gamma q_t^\top k_{v,s}\right)}, \tag{8}$$

where $\mathcal{P}$ contains positions with valid step targets and γ is a learned logit scale. Treating the other steps from the same video as negatives encourages sensitivity to both instruction semantics and procedural position. Since <|STEP|> precedes <|FLAG|>, this plan-aware representation is available to the onset gate and response decoder. Projection details, target construction, and optimzation related details are provided in Appendix G.1.

4.5 TRAINING OBJECTIVE

We train the model with three objectives:

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{main}}}_{\text{what to say, and when to stop}} + \lambda_a \underbrace{\mathcal{L}_{\text{onset}}}_{\text{when to start}} + \lambda_s \underbrace{\mathcal{L}_{\text{step}}}_{\text{where in the plan}}. \tag{9}$$

$\mathcal{L}_{\text{main}}$ is the causal language-modelling loss over all assistant responses, it carries the boundary markers of equation 6. Input tokens and <|silent|> token is masked from this loss.

For onset prediction, we use positive-weighted binary cross-entropy loss over the supervised set $\mathcal{M}$:

$$\mathcal{L}_{\text{point}} = -\frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} \left[\rho y_t \log p_t + (1 - y_t) \log(1 - p_t)\right], \tag{10}$$

where ρ up-weights sparse proactive-onset class. We also regularize the predicted onset frequency:

$$\mathcal{L}_{\text{rate}} = \left(\frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} p_t - \frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} y_t\right)^2. \tag{11}$$

The complete onset objective is

$$\mathcal{L}_{\text{onset}} = \mathcal{L}_{\text{point}} + \lambda_r \mathcal{L}_{\text{rate}}. \tag{12}$$

Because y_t counts proactive onsets rather than all speaking seconds, $\mathcal{L}_{\text{rate}}$ matches the model's onset frequency to the ground truth onset frequency. Thus, the intervention frequency is governed by the gate, while duration is governed independently by the generated boundary markers.

4.6 INFERENCE

At test time the gate and the markers compose into a small state machine. Let ℓ_t be the number of seconds the current utterance has been running and $\hat{m}_{t-1}$ be the marker the model actually emitted at $t-1$. The model speaks at second t iff

$$a_t = \begin{cases} 1, & \hat{m}_{t-1} = \text{<|cont|>} \quad \text{(routed)} \\ \mathbb{I}[p_t \geq \tau], & \text{otherwise} \quad \text{(gated)} \end{cases} \tag{13}$$

The gate is consulted only when no utterance is in progress, exactly the positions it was trained on, and mid-utterance the decision is read off the previous emission. The two mechanisms are therefore used in disjoint states, and neither is asked to model the other's behaviour.

Table 1: **Main set.** Proactivity timing and delivery quality results. Delivery dimensions are LLM-judged. **Bold** marks the best result and underline the second best.

Model	Proactivity			Online Utility			Task Quality			AnswerQ
	F1↑	TimeDiff↓	pAUC↑	Time↑	Rate↑	TextQ↑	Verdict↑	Contin.↑	Complete.↑	Match↑
OFFLINE										
Qwen3-VL	19.00	8.10	22.71	1.40	1.00	3.60	2.90	1.10	3.20	3.31
PROACTIVE										
LiveStar	8.40	21.10	22.28	1.00	1.20	1.40	1.00	1.00	1.00	1.04
MMDuet2	17.70	10.10	15.81	1.10	1.00	1.12	1.00	1.00	1.00	1.36
VideoLLM-Online	33.66	6.25	16.73	3.20	2.96	3.91	2.80	2.86	2.44	2.96
REAL-TIME										
LiveCC-Base	12.23	18.50	22.27	1.70	1.75	4.20	2.95	**3.77**	1.73	1.14
LiveCC-Instruct	18.72	8.40	16.49	1.04	1.00	2.47	1.47	1.10	1.99	1.14
REAL-TIME PROACTIVE										
MageVL	11.39	7.04	19.22	1.08	1.13	2.86	1.25	1.05	1.36	1.23
JoyAI-VL	21.64	6.35	28.90	2.85	2.92	4.50	2.66	2.92	2.55	3.41
Proact-VL	35.04	4.04	37.33	3.23	2.81	4.47	2.84	2.78	4.36	3.44
ONSTEP	**39.70**	**3.43**	**38.03**	**3.50**	**3.87**	**4.74**	**3.12**	3.33	**4.39**	**3.46**

5 EXPERIMENTS AND RESULTS

5.1 SETUP

Datasets. The ProactiveCooking Main split has 213/62/109 train/val/test videos, with 2,524/753/1,489 instruction-feedback and 2,049/575/1,142 question-answer pairs. Advanced, which includes reordered, repeated, and skipped steps, has 209/57/36 videos, 2,495/666/453 instruction-feedback pairs, and 1,889/541/330 question-answer pairs. Both are annotated per second.

Metrics. We evaluate three capabilities. *Proactive timing* uses per-second speaking F1, coverage-penalized onset error (TimeDiff; lower is better), and pAUC (Wang et al., 2025). An LLM judge scores *Online Utility*-timing (Time), frequency (Rate), and utterance quality (TextQ), and *Task Quality* measures feedback correctness (Verdict), coherence (Continuity), and recipe coverage (Completeness), on five-point scales. *Answer quality* (AnswerQ) is the macro-average correctness across six question types. Appendix C defines all metrics and Appendix E reports per-type results.

Baselines. We compare with offline Qwen3-VL; proactive LiveStar (Yang et al., 2025b), MMDuet2 (Wang et al., 2026), and VideoLLM-Online (Chen et al., 2024); real-time LiveCC-Base and LiveCC-Instruct (Chen et al., 2025); and real-time proactive Mage-VL (Yang et al., 2026), JoyAI-VL (Yao et al., 2026), and Proact-VL (Yan et al., 2026).

Implementation details. We fine-tune Qwen3-VL-8B-Instruct for two epochs with learning rate 10^{-5}, cosine decay, bf16, gradient clipping, and FSDP over 8 H100 GPUs. The effective batch size is 64; the vision encoder is frozen. We set $\lambda_a = 0.2$, $\rho = 5$, and $\lambda_s = 0.1$; training takes ~ 70 GPU-hours. We use GPT-5.6-luna for evaluation and Claude-Sonnet-5 for QA generation.

5.2 RESULTS

Proactive timing. Tables 1 and 2 show that ONSTEP achieves the best F1 and TimeDiff on both sets. Against the strongest baseline, Proact-VL, Main-set F1 rises from 35.04 to 39.70 and TimeDiff falls from 4.04 to 3.43 seconds; on Advanced, F1 rises from 32.79 to 37.50 and TimeDiff falls from 4.40 to 3.74 seconds. ONSTEP also improves pAUC from 37.33 to 38.03 on Main and from 33.60 to 34.51 on Advanced.

Delivery and answer quality. On Main, ONSTEP leads five of six delivery dimensions and is second on Continuity, including a large Rate gain over Proact-VL (3.87 versus 2.81). On Advanced, it leads all Online Utility dimensions and ranks second on Verdict and Completeness. These results distinguish fluent output from timely assistance: LiveCC-Base has the best Continuity but only 12.23 F1 and 18.50-second TimeDiff on Main. ONSTEP instead combines the strongest timing with high delivery quality. For interactive QA, ONSTEP scores 3.46 on Main, versus 3.44 for Proact-VL,

Table 2: **Advanced set.** Same protocol and grouping as Table 1. The non-monotonic execution of recipes in this split widen the gap between models that speak selectively and those that do not.

Model	Proactivity			Online Utility			Task Quality			AnswerQ
	F1 ↑	TimeDiff ↓	pAUC ↑	Time ↑	Rate ↑	TextQ ↑	Verdict ↑	Contin. ↑	Compl. ↑	Match ↑
OFFLINE										
Qwen3-VL	22.50	8.02	18.01	1.40	1.00	3.60	1.70	1.10	**2.70**	3.38
PROACTIVE										
LiveStar	10.17	18.73	12.00	1.00	1.10	1.40	1.00	1.00	1.00	1.05
MMDuet2	20.74	9.84	15.24	1.00	1.00	1.00	1.00	1.00	1.00	1.33
VideoLLM-Online	29.00	8.21	11.90	2.29	2.10	3.58	1.40	2.39	1.57	2.88
REAL-TIME										
LiveCC-Base	17.15	15.64	17.41	1.75	1.67	4.30	**2.08**	**2.44**	1.64	1.14
LiveCC-Instruct	22.59	7.59	15.43	1.03	1.00	2.83	1.28	1.11	2.28	1.78
REAL-TIME PROACTIVE										
MageVL	12.87	6.22	17.54	1.08	1.22	2.97	1.17	1.11	1.36	1.22
JoyAI-VL	19.13	8.52	28.92	2.08	2.08	4.25	1.62	2.33	2.22	3.44
Proact-VL	32.79	4.40	33.60	2.30	1.51	3.47	1.80	1.20	2.20	3.53
ONSTEP	**37.50**	**3.74**	**34.51**	**2.67**	**2.92**	**4.32**	1.92	1.75	2.42	**3.55**

3.41 for JoyAI-VL, and 3.31 for Qwen3-VL, and also surpasses all baselines on Advanced. Thus its timing gains do not sacrifice the underlying model's ability to answer questions.

Performance under non-monotonic execution. Advanced set requires tracking the performed step rather than the next canonical one. Although denser reference speech can inflate F1 for frequent speakers, ONSTEP remains best on all three proactivity metrics. On this challenging split, ONSTEP achieves the best F1, TimeDiff, and pAUC, leads all three Online Utility dimensions (2.67 Time, 2.92 Rate, and 4.32 TextQ), and ranks second on both Verdict and Completeness. These results indicate that its timing advantage reflects more selective and useful guidance rather than simply a higher speaking rate. Completeness is ambiguous here because correctly declining to reissue an abandoned step may reduce the score.

5.3 ABLATION STUDY

Table 3 adds components cumulatively to the Qwen3-VL backbone. The onset head together with the <|cont|> and <|done|> boundary tokens adapted to the streaming decision problem, provides the largest improvement, increasing F1 from 19.00 to 35.33, reducing TimeDiff from 8.10 to 4.21 seconds, and improving pAUC from 22.71 to 37.10. Because the onset head and boundary tokens are introduced together, this row measures their combined contribution rather than isolating either component.

Table 3: **Ablation** on main set. Components are added cumulatively to the Qwen3-VL backbone.

Configuration	F1 ↑	TimeDiff ↓	pAUC ↑
Qwen3-VL	19.00	8.10	22.71
+ onset head	35.33 +16.33	4.21 −3.89	37.10 +14.40
+ onset rate	38.19 +2.86	3.75 −0.46	37.80 +0.70
+ step loss	**39.70** +1.51	**3.43** −0.32	**38.03** +0.23

The onset regularizer improves F1 by 2.86 points and reduces TimeDiff by (0.46) seconds, while changing pAUC by (0.70). This is expected as matching the predicted and reference onset rates suppresses excessive interventions rather than changing the ranking of candidate moments.

The step-anchored objective contributes 1.51 F1 points and reduces TimeDiff by (0.32) seconds. Although smaller than the gain from onset modelling, this consistent improvement indicates that explicitly representing progress through the recipe helps localize when guidance should be delivered.

6 CONCLUSION

We formulate procedural assistance as a unified streaming task requiring an assistant to decide when to speak, track progress, provide proactive guidance, and answer questions. We introduce ProactiveCooking , with 6,526 verified, temporally grounded interactions, and ONSTEP , which separates utterance onset from continuation and anchors state to the active recipe step. ONSTEP improves timing and delivery while preserving question-answering performance. Future work should improve long-horizon state tracking and the detection of subtle execution errors.

Ethics Statement

This work builds on previously released CaptainCook4D videos and QIC annotations; we collected no new recordings of cooks. The question-answer annotations were generated from those data and verified by 125 annotators, with three independent judgments per retained candidate and majority resolution as described in Section 3.2. We report only aggregate annotator statistics and do not release annotator identities. Any release of derived annotations must follow the licenses and access conditions of the source datasets. The source videos cover a limited set of recipes, environments, and participants, so performance may not transfer equitably across cuisines, accessibility needs, kitchens, or cultural practices. We encourage evaluation with affected users and explicit safety controls before any real-world use.

AI Use Statement

Generative AI tools were used as components of the research methodology. Claude Sonnet 5 generated candidate question-answer interactions for ProactiveCooking ; these candidates were subsequently screened and video-grounded by human reviewers, as described in Section 3.2. GPT-5.6-Luna was used for the LLM-based evaluation metrics, and Claude Sonnet 5 was additionally used as a second judge in the robustness analysis in Appendix E.2. The models, prompts, and evaluation protocols are reported in Section 5.1 and Appendices B and C.

Reproducibility Statement

We provide the dataset sources, splits, generation and human-verification procedure, and summary statistics in Section 3.1 and Appendix D; the complete generation prompts appear in Appendix B. Section 5.1 reports the backbone, optimization settings, loss weights, compute, and evaluation models, while the method section and appendix give the complete step-objective construction. Appendix C defines every metric and includes the full judge prompts, and the ablations and threshold sweep report sensitivity to key design choices.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

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. URL https://arxiv.org/abs/2511.21631.

Yuwei Bao, Keunwoo Peter Yu, Yichi Zhang, Shane Storks, Itamar Bar-Yossef, Alexander De La Iglesia, Megan Su, Xiao-Lin Zheng, and Joyce Chai. Can foundation models watch, talk and guide you step by step to make a cake? In *EMNLP Findings*, 2023.

Apratim Bhattacharyya, Bicheng Xu, Sanjay Haresh, Reza Pourreza, Litian Liu, Sunny Panchal, Leonid Sigal, and Roland Memisevic. Can multi-modal LLMs provide live step-by-step task guidance? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Livecc: Learning video llm with streaming speech transcription at scale. *arXiv preprint arXiv:2504.16030*, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. In *IJCV*, 2022.

Xin Ding, Hao Wu, Yifan Yang, Shiqi Jiang, Donglin Bai, Zhibo Chen, and Ting Cao. Streammind: Unlocking full frame rate streaming video dialogue through event-gated cognition. *arXiv preprint arXiv:2503.06220*, 2025.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, et al. Ego4d: Around the world in 3, 000 hours of egocentric video. In *CVPR*, 2022.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives. *CoRR*, abs/2311.18259, 2023.

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*, 2025a. ISSN 2835-8856. URL https://openreview.net/forum?id=zKv8qULV6n.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

Yifei Li, Junbo Niu, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, et al. Ovo-bench: How far is your video-llms from real-world online video understanding? *arXiv preprint arXiv:2501.05510*, 2025b.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=w0H2xGHlkw.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024a. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Jihao Liu, Zhiding Yu, Shiyi Lan, Shihao Wang, Rongyao Fang, Jan Kautz, Hongsheng Li, and Jose M Alvare. Streamchat: Chatting with streaming video. *arXiv preprint arXiv:2412.08646*, 2024b.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019.

Sunny Panchal, Apratim Bhattacharyya, Guillaume Berger, Antoine Mercier, Cornelius Böhm, Florian Dietrichkeit, Reza Pourreza, Xuanlin Li, Pulkit Madan, Mingu Lee, Mark Todorovich, Ingo Bax, and Roland Memisevic. What to say and when to say it: Live fitness coaching as a testbed for situated interaction. In *NeurIPS*, 2024.

Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Qifan Zhang, Jikai Wang, Vasundhara Komaragiri, Eric D. Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. Captaincook4d: A dataset for understanding errors in procedural activities. In *NeurIPS*, 2024.

Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *CVPR*, 2022.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. COIN: A large-scale dataset for comprehensive instructional video analysis. In *CVPR*, 2019.

Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, Neel Joshi, and Marc Pollefeys. Holoassist: an egocentric human interaction dataset for interactive AI assistants in the real world. In *ICCV*, 2023.

Yueqian Wang, Xiaojun Meng, Yifan Wang, Huishuai Zhang, and Dongyan Zhao. Proactivevideoqa: A comprehensive benchmark evaluating proactive interactions in video large language models. *arXiv preprint arXiv:2507.09313*, 2025.

Yueqian Wang, Songxiang Liu, Disong Wang, Nuo Xu, Wan Guanglu, Huishuai Zhang, and Dongyan Zhao. MMDuet2: Enhancing proactive interaction of video MLLMs with multi-turn reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=gVbPWbA97s.

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 *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=k9PKgV0L4C.

Senqiao Yang, Kaichen Zhang, Zhaoyang Jia, Jinghao Guo, Yifei Shen, Xinjie Zhang, Xiaoyi Zhang, Haoqing Wang, Xiao Li, Peng Zhang, et al. Mage-vl: An efficient codec-native streaming multimodal foundation model. *arXiv preprint arXiv:2607.24904*, 2026.

Zhenyu Yang, Yuhang Hu, Zemin Du, Dizhan Xue, Shengsheng Qian, Jiahong Wu, Fan Yang, Weiming Dong, and Changsheng Xu. Svbench: A benchmark with temporal multi-turn dialogues for streaming video understanding. *arXiv preprint arXiv:2502.10810*, 2025a.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=4n7IifN7yr.

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, and Jiaqi Wang. Joyai-vl-interaction: Real-time vision-language interaction intelligence, 2026. URL https://arxiv.org/abs/2606.14777.

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. *arXiv preprint arXiv:2504.17343*, 2025.

Luowei Zhou, Chenliang Xu, and Jason Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018.

A HUMAN VERIFICATION: DETAILED RESULTS

A.1 WHAT HUMAN VERIFICATION REVEALS ABOUT AUTOMATIC GENERATION

In this section, we report the outcome of the video-grounded verification stage at item level, after two-of-three majority resolution. We break the results down three ways: by dataset split (table 4), by the surface form of the question (table 5), and by the failure mode of the answers that annotators rewrote (table 6).

A.2 OUTCOME BY SPLIT

Table 4: Human verification outcome per split, at item level after majority resolution. *Retained* = accepted + corrected.

Split	Units	Acc.	Corr.	Rej.	Unsure	No maj.	No verd.	**Retained**	Ret. rate
main train	2,186	1,673	376	52	29	54	2	**2,049**	93.7%
main val	677	444	131	14	57	31	0	**575**	84.9%
main test	1,186	955	187	25	0	19	0	**1,142**	96.3%
adv train	1,964	1,682	207	44	0	26	5	**1,889**	96.2%
adv val	571	463	78	17	0	12	1	**541**	94.7%
adv test	358	293	37	17	0	11	0	**330**	92.2%
Total	**6,942**	**5,510**	**1,016**	**169**	**86**	**153**	**8**	**6,526**	**94.0%**

Overall retention is high: 6,526 of 6,942 reviewed units survive, a rate of 94.0%. The more informative figure is how that 6.0% loss divides. Only 169 units (2.4%) were rejected outright, meaning the question itself should never have been asked, while 1,016 (14.6%) were accepted as questions but had their answers rewritten. Corrections therefore outnumber rejections six to one. The generator is reliable at deciding *what is worth asking* at a given point in a recipe, and markedly less reliable at *answering* it, a separation that matters, because the question side is what the insertion policy of section 3.2 controls, whereas the answer side depends on visual grounding.

Retention varies from 84.9% (main val) to 96.3% (main test). The main val figure is not driven by poor questions: that split's rejection rate, 2.1%, is among the lowest. It is driven by the *unsure* verdict, which accounts for 8.4% of the split. All 86 unsure verdicts in the entire collection fall in main train and main val and none in the four remaining splits, so this reflects a change in the annotation interface partway through collection rather than a property of the data; we retain the label rather than silently recoding it. The highest rejection rate, 4.8%, is on the advanced test split, roughly twice the pooled rate, consistent with the re-issued-instruction structure of the Advanced Planning set making step boundaries harder to pin down.

A.3 OUTCOME BY QUESTION FORM

Table 5 groups items by the surface form of the question, which is a coarser partition than the generator's own taxonomy (table 8). The dominant pattern is that **localisation is the failure mode, not perception in general**. *Where* questions are 28.0% of all items but have an accept rate of 53.8%, the lowest of any form and more than thirteen points below the next lowest (*Which*, 67.0%). By contrast *how many / how much* questions, which are equally dependent on the frames, reach 91.4%, a gap of 37.7 points between counting and locating. Recognising that an object is present, and how many of it there are, is close to solved at this resolution; saying where it is, is not.

The two forms that require no visual grounding at all behave exactly as their construction predicts. *Steps remaining* (785 items), whose answer is computed from the parsed plan, and *Why* (215 items), which draws on culinary world knowledge rather than the footage, together account for 1,000 items with **zero corrections and zero rejections** between them. Their accept rates fall below 100% only through unsure and no-majority verdicts, never through a wrong answer. This is direct evidence for the design choice of section 3.1: routing an answer to a named, checkable source removes the error mode entirely rather than merely reducing it.

Table 5: Question mix in the verified set and item-level accept rate by surface form. The human verification export records the surface form of each question; the generator's own type label of Table 8 is retained separately in the released data.

Surface form	Items	Share	Accepted	Corrected	Rejected	Accept rate
Where (spatial grounding)	1,945	28.0%	1,046	753	72	53.8%
How many / how much	1,939	27.9%	1,773	64	59	91.4%
Yes/No	1,175	16.9%	1,054	76	17	89.7%
Steps remaining	785	11.3%	701	0	0	89.3%
Which	315	4.5%	211	87	10	67.0%
What	224	3.2%	175	32	10	78.1%
How long / how far	218	3.1%	211	3	1	96.8%
Why	215	3.1%	215	0	0	100.0%
How (other)	98	1.4%	97	1	0	99.0%
Other	28	0.4%	27	0	0	96.4%
All	**6,942**	**100%**	**5,510**	**1,016**	**169**	**79.4%**

A.4 What annotators corrected

Table 6: Failure modes among the 1,016 corrected items. Grouping is lexical; the theme boundaries should be read as approximate and the counts within them as exact.

Failure mode	Items	Share	Concentrated in
Wrong location	618	60.8%	Where
Other rewording or factual fix	215	21.2%	Which, Yes/No, How many
Asserted not visible; annotator found it	137	13.5%	Where
Wrong quantity or number	30	3.0%	How many / how much
Answer materially expanded	9	0.9%	Yes/No, What, Which
Fabricated countdown to completion	7	0.7%	Yes/No, How many

Table 6 decomposes the 1,016 rewritten answers. Spatial claims dominate: *wrong location* (618) and *asserted not visible; annotator found it* (137) together make up 74.3% of every correction in the dataset. Set against table 5, 753 of the 1,016 corrections, 74.1% attach to *Where* questions, which are only 28.0% of items. A single capability accounts for three-quarters of the annotation effort.

The second spatial mode is worth separating from the first. In 137 cases the model declined to answer, asserting the object was not visible, and the annotator located it in the frame. This is a false negative rather than a fabrication, and is the safer of the two behaviours: an assistant that admits it cannot see something is more recoverable than one that confidently names the wrong place. The genuinely fabricated mode, an invented countdown to completion, appears in only 7 items (0.7%), which is consistent with timing answers being derived rather than generated. Quantity errors are similarly rare at 30 items (3.0%), matching the high accept rate of counting questions in table 5.

Of the 5,068 items that received a timing judgement, only 3,022 (59.6%) were judged well timed; 1,675 (33.1%) were too early and 371 (7.3%) too late. The skew toward earliness follows directly from the placement rule of Section 3.2: asking at the start of the action avoids questions about work already done, but frequently asks before the object in question has come into view.

Taken together, the three tables support a single conclusion: on this benchmark the bottleneck in automatically generated interactive QA is not question selection, topical relevance, or counting, but spatial grounding, and the question types whose answers are derived from the plan or from world knowledge, rather than from the frames, require almost no correction.

A.5 Text-Only Rejections

Before video-grounded annotation, every generated question–answer pair was screened by a single reviewer who saw only the text of the pair, the instruction being taught at that moment, and the object inventory extracted from the sampled frames, and issued a binary keep/reject verdict. Of the $9{,}486$

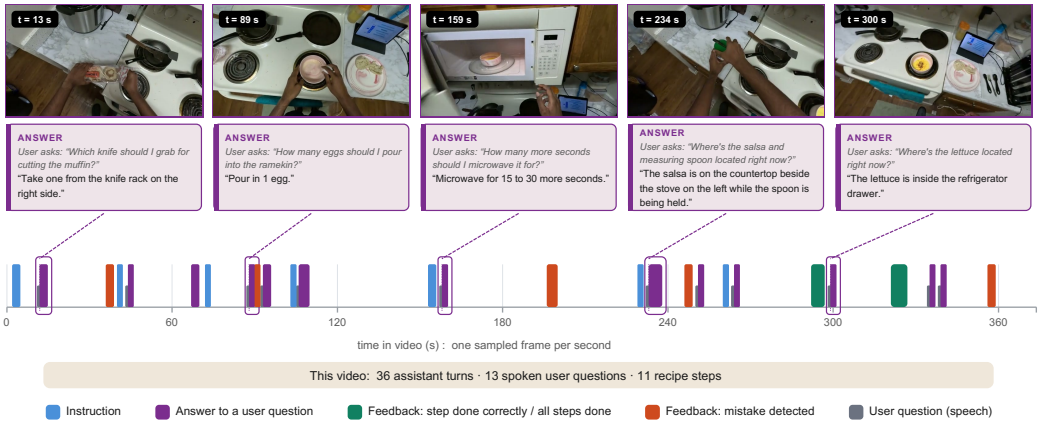


Figure 4: egg and English muffin sandwich

generated candidates, an automatic pre-filter removed 1,727 (18.2%), chiefly yes/no state questions; the remaining 7,759 were reviewed by hand, of which 6,991 (90.1%) were accepted and 768 (9.9%) rejected, at a slightly higher rate in the Advanced split (11.0%) than in the Main split (9.1%).

Rejections fell into three classes. Most (474; 61.7%) were progress questions, such as *Is the oil hot enough yet?*, answered with a remaining-time estimate. These were typically rejected because the same progress question had been inserted multiple times within a single working interval, often at timestamps separated by only a few seconds. A representative example is:

> **Instruction:** Microwave the ramen for 4 minutes.
> **Question:** How much time is left on the microwave?
> **Answer:** Getting there—about 30 more seconds.
> **Rejection reason:** A near-identical question had already appeared in the same working interval.

Such questions came almost entirely from one generator intent, whose reject rate was 60.4%, compared with at most 6.6% for every other intent. A further 215 (28.0%) were redundant, repeating verbatim or near-verbatim a question already asked elsewhere in the same video, a consequence of sampling each step independently; the reviewer retained the first instance and dropped later ones.

The remaining 79 (10.3%) were item-specific problems, including questions about a referent unrelated to the current instruction, premises that conflicted with the step, over-specified spatial questions, and questions that were pragmatically unnecessary given the observed context. For example:

> **Instruction:** Add $1/2$ tbsp of softened butter.
> **Question:** Which measuring spoon should I use for the butter?
> **Answer:** Use the red measuring spoon in your right hand.
> **Rejection reason:** The user was already holding the referenced spoon, making the question unnecessary in context.

A.6 Training Samples

Figure 4, 5, and 6 shows the query-answer examples from ProactiveCooking . Each session is an egocentric recording of one recipe, annotated second by second with everything the assistant should say: the next instruction, feedback on whether the step just taken was correct or mistaken. Here, we focus on the interaction(question-answer) between the user and the assistant. The first two panels show single sessions on a shared time axis, with query/answer pairs set against the frame the user was looking at when they asked. In total the dataset covers 384 distinct videos and 94 hours of video, carrying 9,365 instructions, 9,038 feedback turns and 6,526 distinct user queries, every one of which is answered in place and annotated as its own timestamped turn. Queries are not incidental to the recordings: a session contains between 3 and 42 of them, median 16, so on the order of one a minute. The same recipes are annotated under two regimes, a main split in which the steps are

followed in order, and an advanced split in which they are not, which lets the same question be asked of a system that can no longer assume the user is where the recipe says they are.

B Question Generation Prompts

Prompt A: visual evidence (multimodal; once per interval)

```
You are a meticulous kitchen vision annotator. Report ONLY what is actually visible in
the provided frames; never guess.
You are given {N} sampled video frames, in order.
[if no frames: ''No video frames are available; do not invent visual details.'']
Instructor's latest guidance (the user is performing this now): {instruction}
Identify the objects, actions, and surfaces yourself from the frames; do not rely on any
predefined list.
Return ONLY a JSON object with these keys:
- visible_objects: list of {name, location, confidence} for objects you can clearly see;
location is a short phrase, confidence is high/medium/low.
- visible_actions: list of short phrases for actions visibly happening.
- visible_surfaces: list of surfaces/containers visibly present.
- uncertain_objects: list of objects the instruction implies but you cannot confirm from
the frames.
- caption: one sentence describing the frames.
If no frames are available, return empty lists for every key except a brief caption noting
that no frames were seen.
Example of the required shape:
{"visible_objects": [{"name": "butter knife", "location": "on the cutting board",
"confidence": "high"}], "visible_actions": ["spreading nut butter on the tortilla"],
"visible_surfaces": ["wooden cutting board"], "uncertain_objects": ["jelly"], "caption":
"A hand spreads nut butter onto a tortilla on a cutting board."}
```

Stage 1 of the pipeline (section 3.2) issues **two** model calls per working interval. The first converts sampled frames into a structured description of the scene; the second writes a question-answer pair conditioned on that description together with the recipe. Splitting the work this way is deliberate: it forces the visual claim underlying every question to be written down explicitly, so a wrong answer can be traced to either a misperception (Prompt A) or a reasoning failure (Prompt B), and the intermediate evidence is auditable.

Neither call sees the video directly at generation time. Prompt A is the only multimodal call in the pipeline; Prompt B receives its output as JSON text. This also bounds cost, since Prompt A runs once per interval regardless of how many question types are attempted.

Prompt A: visual evidence. Frames sampled from the opening of the working interval are summarised into objects with an approximate location and a confidence rating, actions under way, surfaces in view, objects the instruction implies but that cannot be confirmed, and a one-sentence caption (section B). Two properties matter downstream. The `uncertain_objects` list is what later licenses an identification question ("is the red spread jelly or sauce?") rather than a location question, and the explicit no-frames branch makes the degenerate case safe: with no frames the model is told to return empty lists rather than speculate, which is what keeps the ungrounded question type usable when the visual path fails.

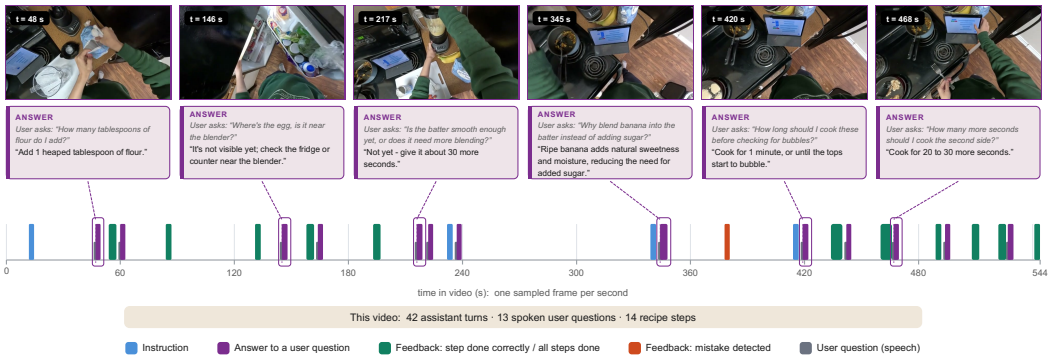


Figure 5: banana and strawberry pancakes

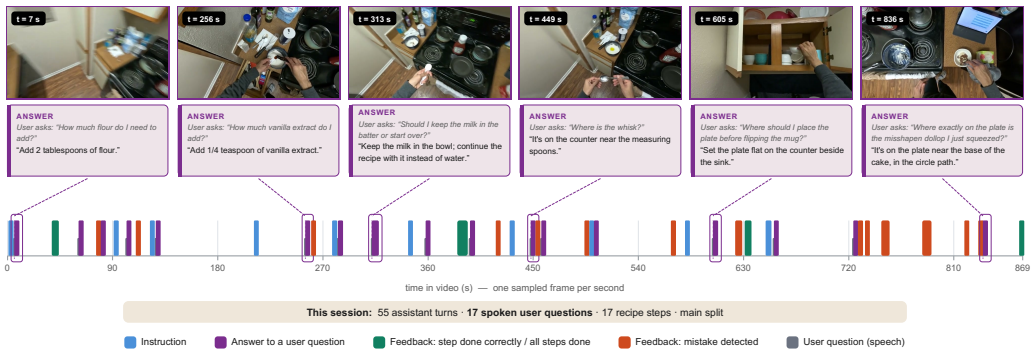


Figure 6: frosted mug cake

Prompt B: question and answer generation. Framing and constraints vary by type; the TASK block is spliced in from Figure B.

Framing (one of three, by type):
default: The user has just heard the current step and is starting it. As they begin, they pause to ask the instructor ONE question about what they are about to do. Write that question and the instructor's real-time answer.
mid_step_state: The user is partway through the current step and pauses to ask the instructor ONE question. Write ONLY that question; its answer is filled in separately.
keyword_knowledge: The user is about to use the main ingredient/tool of the current step and pauses to ask the instructor ONE general question about it. Write that question and the instructor's short, factual answer.

TASK: {per-type instructions, Figure B}

Constraints (one of three, by type):
default: The question MUST be something a cook would genuinely ask WHILE cooking to make progress on the CURRENT step - never a trivial or decorative detail (the color, shape, brand, or material of an object) that does not affect the cooking. The answer MUST be OBJECTIVE and verifiable - a value, a number, a place, or one of a fixed set - never a matter of opinion and never a plain yes/no. Do NOT ask whether the user is doing something correctly/properly/well, and do NOT judge technique or appearance. Do NOT ask where an object is if it is already in the cook's hands or clearly in front of them - only ask about things they must find or check. This is a live, spoken interaction, so keep both extremely short: question under 20 words, answer under 20 words, one plain sentence each, no preamble, and do not restate the question.
mid_step_state: This is a live, spoken interaction: keep the question extremely short (under 20 words), one plain sentence, no preamble. Leave answer empty.
keyword_knowledge: Ask about a GENERALLY-KNOWN culinary property of the ingredient/tool - its taste, heat, typical ingredients, texture, or a common substitute. A short yes/no answer with a brief reason is fine. Do NOT ask about this specific brand/jar, exact amounts, or anything only the video could answer. Keep both short: question under 20 words, answer under 20 words, one plain sentence each, no preamble, and do not restate the question.

Instructor's latest guidance (current step): {instruction}
Measurements in this step: {comma-separated values, or (none)}
Full recipe (current step marked):
1. {step 1}
...
k. {step k} <-- CURRENT STEP
...
Visual evidence (JSON): {output of Prompt A}

If you cannot ask a question of the requested kind, return exactly {"skip": true, "reason": "..."}.
Return ONLY a JSON object with keys: skip(false), grounding_type, question, answer, evidence_object. Example:
{"skip": false, "grounding_type": "keyword_focus", "question": "How many onions are on the board?", "answer": "There are two onions on the board.", "evidence_object": "onion"}

Prompt B: question-answer generation. The second call assembles four parts: a framing sentence placing the user at the right moment of the step, a per-type TASK block, a set of constraints, and the context (current instruction, its measurements, the full recipe with the current step marked, and the Prompt A evidence as JSON). It returns a JSON object, or {"skip": true} when no question of the requested kind can be grounded, an explicit refusal channel rather than a forced generation, which is what keeps the type mix honest when a type is infeasible at that interval.

Three constraint sets exist, selected by type. The default set is the strictest: it requires an objective, verifiable answer, bans yes/no answers and opinion, forbids questions about decorative attributes (colour, shape, brand, material) that do not affect cooking, and forbids asking where an object is

when it is already in the cook's hands. The `mid_step_state` set asks for the question only and leaves the answer empty, because that answer is computed from the interval timing rather than generated. The `keyword_knowledge` set inverts the grounding requirement: it explicitly permits a yes/no answer and restricts the question to generally-known culinary properties, since by design that type is answered from world knowledge and cannot be checked against frames.

All prompts request answers under twenty words in a single plain sentence. This is not a formatting preference: the target is speech delivered to someone whose hands are busy, and the same brevity requirement is what the Rate dimension scores.

Per-type TASK blocks spliced into Prompt B

```
instruction_spec: Ask about a SPECIFIC VALUE stated in the current instruction - a
quantity, count, size, spacing, thickness, or time (how much, how many, how thick, how far
apart, how long). The answer MUST be that exact value from the instruction. Use
grounding_type "instruction_spec".
keyword_focus: First identify the MAIN object of the current instruction - the thing being
acted on (e.g. 'onion' in 'chop the onion'). Then ask ONE factual VISUAL question about
THAT object whose answer requires looking at the frames: its location ('Where is the
onion?') or a count ('How many onions are there?'). The answer states that visual fact
(a place or a number). Use grounding_type "keyword_focus". Only ask WHERE the object is
if the cook would actually need to find it to start the step - NOT when it is already in
their hands or clearly in front of them (never produce answers like 'it is in your
hands'). [appended when identification is allowed:] If the main object is genuinely
ambiguous (listed in uncertain_objects), you may instead ask which-one it is (e.g. 'Is the
red spread jelly or sauce?'); the answer names the identity.
keyword_knowledge: First identify the MAIN object of the current instruction - the
ingredient or tool being used (e.g. 'chilli sauce' in 'add the chilli sauce'). Then ask
ONE general-interest CULINARY-KNOWLEDGE question about it that a curious cook might ask -
its typical taste, common ingredients, spiciness/heat, texture, or a common substitute
(e.g. 'Is chilli sauce very spicy?', 'Does chilli sauce usually contain black pepper?').
Answer from GENERAL cooking knowledge in one short sentence; a yes/no plus a brief reason
is fine. Ask ONLY about generally-known properties of the ingredient - NOT about this
specific brand/jar, exact quantities, or anything only the video could show. Use
grounding_type "keyword_knowledge".
mid_step_state: Ask ONE short question about the STATE or PROGRESS of the current step's
food/item - e.g. 'Is the toast ready yet?' or 'How much longer until it's done?'. Ask
ONLY the question about doneness/progress; leave the answer empty. Use grounding_type
"mid_step_state".
other_visual: Ask ONE practical VISUAL question that a cook would genuinely ask to carry
out the CURRENT step - e.g. the position or arrangement of the items needed for this step,
whether the required tool/ingredient is within reach, or the state of the work area
relevant to the step. It must be answerable from the frames, must NOT be a numeric spec
from the instruction, and must NOT be the main-object location/count already covered. Do
NOT ask about color, shape, brand, material, or other decorative details - those do not
help with cooking. The answer states a concrete, cooking-relevant visible fact. Use
grounding_type "other_visual".
```

C Evaluation Metrics

A proactive assistant must decide *when* to speak and *what* to say, and the two failure modes are independent: a model can be perfectly timed and factually wrong, or say the right thing several seconds too late. We therefore report six families of metrics. Three are reference-based and exactly reproducible (Sec. C.1-C.2); three are LLM-judged and are reported with their judge configuration, because the judging API exposes no seed (Sec. C.3–C.5).

Throughout, a video is a sequence of seconds $[\text{begin}, \text{end})$. The reference annotates a set of *utterances*; each utterance spans one or more consecutive seconds and is stored as per-second fragments (e.g. `"Use an 8"` / `"inch flour tortilla."`), which we regroup by utterance index before any content-level comparison.

C.1 F1 over speaking seconds

The coarsest question is whether the assistant was talking at the right times. Let P be the set of seconds at which the model emits non-empty text and L the set of seconds at which the reference does. Then

$$\text{Precision} = \frac{|P \cap L|}{|P|}, \qquad \text{Recall} = \frac{|P \cap L|}{|L|}, \qquad \text{F1} = \frac{2\,\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

We report the *micro* average, pooling $|P \cap L|$, $|P|$ and $|L|$ over all videos, so long videos are weighted by their length. Since, F1 measures *overlap of speaking time only*; it is blind to what was said.

C.2 TIMEDIFF

F1 says nothing about *latency*. TimeDiff measures, per reference utterance, how long the assistant took to respond.

Let the i-th reference utterance occupy $[a_i, b_i]$, let t_k be the seconds at which the model speaks, and let $K_i = \{k : t_k \in [a_i, b_i]\}$ be the predictions falling inside it. Define the tolerance window $D_i = [a_i - \delta,\ b_i + \delta]$ and let P_i be the predictions assigned to utterance i (each prediction is assigned to the utterance whose onset it is nearest, so every prediction is charged exactly once). Then

$$\text{TimeDiff}_{\text{orig}}(i) = \begin{cases} \min_{k \in K_i} (t_k - a_i) & K_i \neq \emptyset \\ b_i - a_i & \text{otherwise,} \end{cases}$$

$$\text{TimeDiff}(i) = \text{TimeDiff}_{\text{orig}}(i) + \alpha \left| \{k \in P_i : t_k \notin D_i\} \right|,$$

with $\delta = 2\,\text{s}$ and $\alpha = 0.5$. The second term penalises predictions attributed to an utterance but landing outside its tolerance window. Lower is better.

Coverage. TimeDiff is averaged only over utterances the model responded to at all; an ignored utterance contributes the fallback $b_i - a_i$, which averages $2.34\,\text{s}$ on our data. A near-silent model is therefore charged little and collects no penalty, so TimeDiff *alone* rewards silence. We report alongside it

$$\text{Coverage} = \frac{|\{i : K_i \neq \emptyset\}|}{N},$$

the fraction of reference utterances that received any response.

Combined. Because the two move in opposite directions we report

$$\text{Combined} = \frac{\text{TimeDiff}}{\text{Coverage}},$$

the latency cost per utterance actually answered. Ignoring an utterance can no longer be cheaper than answering it imperfectly. It has the same units as TimeDiff, lower is better, and introduces no tunable constant.

pAUC judge: system prompt (cooking rubric)

```
You are an evaluator for a hands-free cooking assistant that guides a user through a
recipe in real time.  Rate how well the predicted utterance covers the key content of the
ground-truth utterance for this moment.
The assistant does three things:  announce the next recipe step, tell the user whether the
step they just attempted was done correctly, and answer questions.
Use the following scale:
- 3:  Mostly covered; conveys the same instruction, verdict or answer as the ground truth,
with at most minor rephrasing.
- 2:  Partially covered; some correct content, but a significant part of the ground truth
is missing or inaccurate.
- 1:  Incorrect; largely unrelated, missing, or of very poor quality.
Score 1 when the prediction INVERTS a verdict -- calling a mistake a success or inventing
an error the user did not make.  Telling the user a failed step succeeded is the worst
error this assistant can make, and it must not score above 1 however fluent the wording.
This is an instructor, not entertainment:  chatter, filler and commentary for its own sake
add no coverage and should not raise the score.
Consider similarity between past and current Predicted Answers; repeating content already
said earns no additional credit.
Output the score only, do not add more explanations.
```

pAUC judge: per-call user message

```
Ground Truth Answer:  {gold_text}
Predicted Answer:  {growing_prefix_of_predicted_sentences}
```

C.3 PAUC (PROGRESSIVE AREA UNDER CURVE)

pAUC asks how much correct content the assistant had delivered *by each moment*, integrated over time. It is LLM-judged and is the only metric that scores content and timing jointly.

One turn is one reference utterance, with that utterance's own time span as the turn window. We score **every reference utterance as its own turn**. For each turn, predictions falling inside the turn

window are merged into utterances and presented to the judge as a growing prefix, one call per predicted utterance. The judge rates coverage of that turn's reference utterance on $\{1, 2, 3\}$, mapped to $y \in \{0, 1, 2\}$. The resulting points (x_k, y_k), with x_k the offset from the turn onset, define a right-continuous step function that holds y_k until the next prediction. With S the turn span, $y_{\max} = 2$ and an initial value $y_0 = 0.5$ before the model speaks,

$$\text{pAUC} = \frac{1}{S\, y_{\max}} \int_0^S y(x)\, \mathrm{d}x.$$

Online Utility judge: delivery rubric (cooking). Scores Time, Rate and TextU.

```
Task Description:
You are an expert in evaluating the **delivery** of a real-time, second-by-second cooking
assistant that guides a user through a recipe hands-free.

You will be provided:
1) Context Text (background; may be empty)
2) Label Timeline Text (time-aligned reference; [ASSISTANT] = the reference instructor,
[USER] = the user's spoken questions)
3) Prediction Timeline Text (the assistant output to be evaluated)

What to score:
- ONLY score the content in Prediction Timeline Text.
- Context and Label are reference signals and must NOT be scored.

The assistant's job has three parts, all time-critical:
- Instruct:  announce each recipe step when the user is ready for it.
- Verdict:  once a step is attempted, say whether it was done correctly.
- Answer:  respond promptly when [USER] asks a question.

Important notes:
- The assistant is event-triggered:  it speaks only on some seconds; missing seconds are
silence, which is usually CORRECT. Long silences mid-step are expected and must not be
penalized on their own.
- Label Timeline is a reference, not a script.  No verbatim match is required.
- This is a hands-free instructor, NOT entertainment.  Chatter, hype, filler and
commentary-for-its-own-sake are DEFECTS: the user's hands are busy and every spoken second
costs attention.  Silence is the correct default.

Scoring rules:
- Output integer scores in [1..5] (no 0).
- Score bands:  5 excellent; 4 good; 3 mixed; 2 very poor; 1 unusable.

Dimension 1.  Time (does it speak at the right moments):
- 5:  Instructions arrive when the step is due; verdicts follow the user's action
promptly; [USER] questions answered without delay; silent otherwise.
- 4:  Generally well timed; minor drift or a small gap.
- 3:  Noticeably early or late; misses some moments; speaks when silence was right.
- 2:  Frequently misses obvious moments, or speaks over the user.
- 1:  Timing bears no relation to what is happening.
Tolerance:  a response a few seconds either side of the reference window is fine; exact
boundaries are not required.  Do not penalize EARLY responses more than late ones of the
same size.

Dimension 2.  Rate (density and pacing):
- 5:  One idea at a time, short spoken bursts, no dumping a whole recipe at once; no
filler; nothing repeated.
- 4:  Mostly well paced; occasional long burst or mild redundancy.
- 3:  Multiple dense dumps, repetition, or fragmented delivery.
- 2:  Extremely long/dense output, or so sparse that steps go unspoken.
- 1:  Unusable pacing.

Dimension 3.  TextU (clarity as spoken instruction):
- 5:  Natural, unambiguous, immediately actionable by someone not looking at a screen;
quantities and objects stated clearly.
- 4:  Mostly clear; minor awkwardness or fragmentation.
- 3:  Frequently awkward or incomplete; the user would have to guess.
- 2:  Often hard to act on.
- 1:  Incoherent.
STRICT output format (no extra characters):
Time:  <1-5>
Rate:  <1-5>
TextU: <1-5>
Context:  {context}
Label Timeline:  {label_timeline}
Prediction Timeline:  {prediction_timeline}
```

C.4 Online Utility: delivery quality

Online Utility judges *delivery*, whether the assistant spoke at the right moments, at the right density, in language a user with busy hands could act on. The judge receives the reference timeline as a signal and scores only the prediction timeline, on three integer dimensions in $[1, 5]$:

- **Time**: instructions arrive when the step is due, verdicts follow the user's action promptly, and user questions are answered without delay, with silence otherwise. A response a few

seconds either side of the reference window is not penalised, and early responses are not penalised more heavily than late ones of the same size.

- **Rate**: one idea at a time, short spoken bursts, no dumping the recipe at once, no filler, no repetition. Both extremes are defects: dense dumps and so-sparse-that-steps-go-unspoken.
- **TextU**: clarity as *spoken* instruction, unambiguous and immediately actionable without looking at a screen, with quantities and objects stated explicitly.

The rubric states explicitly that this is a hands-free instructor and not entertainment: chatter, hype and commentary-for-its-own-sake are defects, and silence is the correct default.

Task Quality judge: content rubric (cooking). Scores Verdict, Continuity and Completeness.

```
Task Description:
You are an expert in evaluating the **content** of a cooking assistant that guided a user
through a recipe hands-free.
You will be provided:
1) Context Text (background; may be empty)
2) Label Final Text (reference transcript of the whole session)
3) Prediction Text (the assistant output to be evaluated)
What to score:
- ONLY score the content in Prediction Text.
- Context and Label are reference signals and must NOT be scored.
The reference contains three kinds of assistant turn:  recipe INSTRUCTIONS, VERDICTS on
the user's attempt at a step (success, or a mistake of a particular kind -- technique,
measurement, preparation, timing, temperature), and ANSWERS to the user's questions.
Scoring rules:
- Output integer scores in [1..5] (no 0).
- Score bands:  5 excellent; 4 good; 3 mixed; 2 very poor; 1 unusable.
Dimension 1.  Verdict (is the assessment of the user's actions correct):
This is the highest-stakes dimension.  Compare each verdict in Prediction against the
corresponding one in Label.
- 5:  Every verdict matches:  successes confirmed, mistakes caught, and the KIND of
mistake correctly identified.
- 4:  Verdicts almost always correct; at most a misnaming of which kind of mistake.
- 3:  Several verdicts wrong or vague; the user would sometimes be misinformed.
- 2:  Verdicts frequently inverted -- praising failed steps or inventing errors.
- 1:  Verdicts are unrelated to what the user did.
Weighting:  calling a MISTAKE a success is worse than the reverse.  The first lets a
ruined dish proceed; the second is merely irritating.
Dimension 2.  Continuity (coherence across the session):
- 5:  Steps follow logically; consistent references to ingredients and tools; no
repetition of completed steps; the user's thread is maintained.
- 4:  Mostly consistent; minor awkward transitions.
- 3:  Noticeable drift, confusing referents, or repeats already-finished steps.
- 2:  Contradicts earlier turns or loops heavily.
- 1:  No coherent thread.
Dimension 3.  Completeness (does it actually get the user through the recipe):
- 5:  Every recipe step covered, in order, none skipped or invented; every [USER] question
answered.
- 4:  Covers the recipe; a minor step under-specified or a question thinly answered.
- 3:  Skips steps, reorders them, or leaves questions unanswered.
- 2:  Large parts of the recipe missing, or invented steps not in the recipe.
- 1:  Does not guide the user through the task at all.
STRICT output format (no extra characters):
Verdict:  <1-5>
Continuity:  <1-5>
Completeness:  <1-5>
Context:  {context}
Label Final:  {label_final}
Prediction:  {prediction}
```

C.5 Task Quality: session-level correctness

FinalQ judges the session as a whole, on the same $[1, 5]$ integer scale:

- **Verdict** (highest stakes): is the assessment of the user's actions correct successes confirmed, mistakes caught, and the *kind* of mistake correctly identified. The rubric weights calling a mistake a success as worse than the reverse: the first lets a ruined dish proceed, the second is merely irritating.
- **Continuity**: coherence across the session, steps follow logically, references to ingredients and tools stay consistent, and completed steps are not repeated.
- **Completeness**: whether the recipe was actually carried through, as opposed to steps being skipped or the session abandoned.

AnswerQ judge: system prompt

```
You are grading a hands-free cooking assistant that answers questions
a cook asks while their hands are busy.

You will be given the cook's QUESTION, the REFERENCE answer written by
an expert annotator, and the assistant's PREDICTED answer.  Rate how
well the PREDICTED answer conveys the information the REFERENCE does,
from 1 to 5:

5 - Perfect.  Conveys everything the reference does.  A paraphrase
carrying the same facts scores 5; wording, length and politeness are
irrelevant.
4 - Nearly right.  All the important content is there, with a minor
omission or an imprecision that would not mislead the cook.
3 - Partly right.  Some correct content, but a significant part of the
reference is missing, or the answer is vague where the reference is
specific.  Vague-but-not-wrong belongs here, not lower.
2 - Mostly wrong.  Little correct content, or it addresses the
question only glancingly.
1 - Completely wrong.  It contradicts the reference on a concrete
detail -- a size, temperature, duration, quantity, location,
ingredient or tool -- or never addresses the question at all.
Confidently stating a wrong fact is a 1 however fluent it is, and
scores below a vague answer that avoids the claim.

The reference is the ground truth; disagreement with it is the
prediction's error, never the reference's.  The prediction may be a
transcript fragment that runs on into unrelated commentary -- grade
the part that addresses the question and do not penalise it merely for
saying more.

Reply with exactly two lines and nothing else:
SCORE: <a single digit 1-5>
REASON: <one short sentence>
```

AnswerQ judge: per-item user message

```
QUESTION: {question}
REFERENCE: {reference}
PREDICTED: {predicted}
```

C.6 Answer Quality(AnswerQ): answer correctness

AnswerQ asks whether the assistant actually answered the question it was asked. It is the only metric that scores answer *content*. F1 and TimeDiff score only *when* the model spoke. References the model never responded to are scored at the floor rather than dropped, so silence cannot raise the score. The judge rates the prediction on an integer scale in $[1, 5]$.

D Data Preprocessing

D.1 Source Data

We construct our training and evaluation corpus from the Qualcomm Interactive Cooking Bhattacharyya et al. (2025) annotations over the Captain Cook 4D Peddi et al. (2024) egocentric (GoPro) cooking videos, processed at 360p. Each source video is annotated as a temporally ordered sequence of *events* of two types: **instructions**, which specify the next step the user should perform, and **feedback**, which states whether the just-completed step was executed correctly and, when applicable, identifies specific errors in preparation, technique, measurement, temperature, or timing. Each video additionally carries a `remaining_plan` field enumerating the full ordered recipe. A defining property of the source is that events are annotated as *point* timestamps rather than spoken intervals; several of the design choices below follow directly from this. In our target representation of streaming conversational format, in which a video is encoded as a per-second sequence of (visual-chunk, assistant) turns and the model must decide, at every second, whether to remain silent or to emit an utterance.

D.2 INSTRUCTION-FEEDBACK PAIRING

We first group the raw event stream into instruction-feedback *pairs*. Each instruction is matched with the immediately following feedback event; unpaired feedback events (e.g. a terminal "all steps complete" acknowledgement) are retained as feedback-only pairs. Each pair $\pi = (t_{\text{ins}}, s_{\text{ins}}, t_{\text{fb}}, s_{\text{fb}}, \rho)$ records the instruction onset t_{ins}, instruction text s_{ins}, feedback timestamp t_{fb}, feedback text s_{fb}, and the recipe plan ρ available at that point. Let $\Pi = (\pi_1, \ldots, \pi_M)$ denote the ordered pair sequence of a video.

D.3 PER-SECOND CAPTION SYNTHESIS

Because source events are point-timestamped, we synthesize realistic per-second caption for each utterance. We first determine an utterance's spoken duration from a fixed speaking rate of $\omega = 5$ words/second, capped at $U_{\max} = 6$ seconds so that long sentences pack densely rather than stretching unrealistically:

$$t \;=\; \min\Big(\,\lceil n/\omega \rceil,\; U_{\max}\Big), \tag{14}$$

where n is the utterance word count. We then distribute the n words over the t one-second slots using the ProAct-VL Yan et al. (2026) even-distribution rule. With

$$q \;=\; \left\lfloor \tfrac{n}{t} \right\rfloor, \qquad r \;=\; n \bmod t, \tag{15}$$

the first r seconds each receive $q{+}1$ words and the remaining $t{-}r$ seconds each receive q words, preserving word order.

To place utterances on the clip timeline we use an interval layout anchored on the ground-truth timestamps . Each instruction is spread *forward* from its true onset, while the corresponding feedback is spread *backward* so that it terminates exactly at the step's completion timestamp t_{fb}, occupying the otherwise-silent tail of the action. A monotonically non-decreasing cursor c guarantees that captions never overlap; when the available gap between an instruction's end and the completion timestamp is shorter than the feedback's natural length, the feedback is compressed to fit.

D.4 CLIP CONSTRUCTION

Training and validation. We segment each video into overlapping, variable-length clips using a sliding window over instruction-feedback pairs, with window size $P = 5$ pairs and stride $\sigma = 1$ pair . Each clip is padded by $\delta = 2$ seconds of visual context before its first event and after its last. Clips whose duration exceeds $D_{\max} = 90$ s are recursively bisected at the pair level until each fragment fits the cap; clips shorter than $D_{\min} = 10$s are discarded. The dense overlap induced by $\sigma = 1$ ensures that any discarded short fragment remains covered by neighbouring windows.

Test. Segmenting a video into short windows does not reflect the model's test setting, in which it must guide an entire recipe end-to-end as a continuous stream. We therefore emit the test split as a *single uncapped video*, spanning second 0 to the full video duration, with no pair windowing and no maximum-duration cap. This yields a faithful streaming-evaluation protocol; the model's bounded-memory key-value (KV) cache eviction makes processing these long clips tractable at inference time.

D.5 CONDITIONING: RECIPE

Recipe as pinned system context. A natural baseline presents the full recipe as a user query at the start of each clip. We avoid this for two reasons. First, under sliding-window segmentation the query would arrive at an arbitrary, window-dependent second rather than a consistent position. Second, and more importantly, under the model's streaming KV-cache budget the eviction policy preserves the system-prompt tokens but evicts the tokens immediately following them first; a recipe placed in the leading user turn would therefore be the *first* content forgotten on long videos. We instead fold the complete ordered step list into the per-clip system prompt. This (i) yields a clean factorization in which the system prompt specifies the *plan*, the history specifies*progress*, and the video specifies the *current state*; (ii) keeps the recipe *pinned* against cache eviction, so it remains available throughout

Table 7: Cooking dataset statistics after preprocessing. Clip duration and assistant-utterance counts are reported as min / median / max and median, respectively.

Split	Mode	Videos	Clips	Duration (s)	Utt./clip
Train	sliding windows	213	10,014	10 / 66 / 90	4
Val	sliding windows	62	2,984	10 / 68 / 90	4
Test	full-video stream	109	109	239 / 829 / 2,126	63

an entire stream; and (iii) aligns with the instruction-following prior of the Qwen3-VL backbone, which is pretrained to attend to system-role instructions. Consequently our clips contain no user turns: the assistant's guidance objective is conveyed entirely by the system prompt.

Persona augmentation. To prevent the model from overfitting to a single phrasing of the assistant's role, we author six semantically equivalent paraphrases of the cooking instructor persona and sample one uniformly at random per clip; the recipe step list is then appended to the sampled persona. All randomness (persona sampling) is governed by a fixed seed to ensure reproducibility.

D.6 Question Types

We define six question types, each tied to a specific source of evidence against which its answer can be generated or verified. This source-aware design prevents the QA agent from relying on visual evidence when the answer should instead come from the instruction, recipe plan, timestamps, or general culinary knowledge.

Table 8: Interactive question taxonomy and corresponding answer sources.

#	Type	Question target	Answer source
1	`instruction_spec`	Quantity, duration, size, or other value specified by the current instruction	Current instruction
2	`keyword_focus`	Location, identity, or count of the instruction's primary object	Observed video frames
3	`keyword_knowledge`	General culinary knowledge, such as substitutions, flavor, or heat	Model world knowledge
4	`sequential`	Number of recipe steps remaining	Parsed recipe plan
5	`mid_step_state`	Progress or remaining time during a passive cooking step	Question and feedback timestamps
6	`other_visual`	Other step-relevant visual information	Observed video frames

`instruction_spec` questions ask about values explicitly stated in the current instruction, such as quantities, durations, or sizes. Their answers must reproduce the relevant information from the instruction. `keyword_focus` questions ask visually grounded questions about the instruction's primary object, including its location, identity, or count. By contrast, `keyword_knowledge` questions concern general culinary properties of that object, such as suitable substitutions, flavor, or cooking temperature, and are answered using model world knowledge.

Table 9: Comparison to our dataset and benchmark (Ours): where, *Mult-step Goal Driven* refers to whether the videos are driven by a specific goal (eg, cooking a recipe); *Step-by-Step Instructions*: whether the videos contain subjects following a set of step-by-step instructions; *Timed Feedback*: whether the subjects receive timed feedback per step that are sufficient to guide the subjects to the goal; and *Timed Interaction*: whether the subjects engage in timed two-way interaction (eg, asking questions mid-task and receiving answers at the moment they are needed).

Dataset	Domain	Multi-step Goal Driven	Step-by-Step Instructions	Timed Feedback	Timed Interaction	Length (hrs)
Epic-Kitchens Damen et al. (2022)	Cooking	×	×	×	×	100
Ego-4D Grauman et al. (2022)	Daily-life	×	×	×	×	3670
Ego-Exo4D Grauman et al. (2023)	Daily-life	×	×	×	×	1422
Assembly-101 Sener et al. (2022)	Toy Assembly	✓	×	×	×	513
HoloAssist Wang et al. (2023)	Obj. manip.	✓	×	×	×	166
HowTo100M Miech et al. (2019)	Diverse	✓	✓	×	×	134k
COIN Tang et al. (2019)	Diverse	✓	✓	×	×	512
YouCookv2 Zhou et al. (2018)	Cooking	✓	✓	×	×	176
WTAG Bao et al. (2023)	Cooking	✓	✓	×	×	10
QEVD Panchal et al. (2024)	Fitness	×	✓	✓	×	474
QIC Bhattacharyya et al. (2025)	Cooking	✓	✓	✓	×	94
Ours	Cooking	✓	✓	✓	✓	94

The remaining types capture recipe-level and temporal context. `sequential` questions ask how many recipe steps remain, with answers computed from the current position in the parsed recipe plan. `mid_step_state` questions ask about progress or remaining time during passive cooking and waiting steps; their answers are derived from the question and closing-feedback timestamps. Finally, `other_visual` covers step-relevant visual questions that do not fit the narrower `keyword_focus` category.

D.7 Dataset Statistics

Table 7 summarizes the final corpus. The training and validation splits comprise overlapping windowed clips capped at 90 s, whereas the test split comprises one full-length streaming clip per video (up to $\sim$35 minutes). In Table 9, we show the comparison with existing benchmarks.

E Detailed Results

E.1 Question Type Results

Table 10 breaks the score down by question type. The streaming baselines (LiveStar, MMDuet2, LiveCC, MageVL) sit near the floor of the scale in every category, indicating that they rarely produce a usable answer at all, while VideoLLM-Online is competitive only on instruction-specification questions. Among the methods that answer substantively, ONSTEP is the strongest overall: it takes four of the six categories on the advanced set, instruction (4.90), Knowledge (4.34), sequential (2.24) and key-word (3.87), and the best keyword score on the main set knowledge (4.39). The clearest separation is on sequential questions, where every competing real-time proactive system collapses towards the floor (JoyAI-VL $1.43/1.39$, MageVL $1.05/1.05$) while ONSTEP reaches $2.20/2.24$; such questions cannot be answered from the visual stream alone, and so reward a model that has learned to answer rather than only to narrate. ONSTEP also matches or exceeds the offline Qwen3-VL upper bound in five of six advanced categories despite operating under a real-time constraint.

Table 10: Individual scores for each of the question type across the main and advanced set.

Model	Main Set						Advance Set					
	Ins.	Knowledge	Seq.	Mid Step State	keyWord	Visual	Ins.	Knowledge	Seq.	Mid Step State	keyword	Visual
Offline												
Qwen3-VL	4.82	4.04	1.51	3.28	<u>3.62</u>	<u>2.60</u>	4.88	<u>4.33</u>	1.37	<u>3.52</u>	3.82	2.35
Proactive												
LiveStar	1.05	1.07	1.01	1.00	1.03	1.10	1.01	1.00	1.00	1.00	1.20	1.10
MMDuet2	1.54	1.21	1.20	1.31	1.34	1.57	1.29	1.31	1.10	1.28	1.43	1.58
VideoLLM-Online	4.45	3.20	1.64	2.90	3.30	2.30	4.36	3.20	1.63	2.79	3.43	1.90
Real-time												
LiveCC-Base	1.38	1.04	1.00	1.18	1.19	1.07	1.47	1.06	1.00	1.07	1.13	1.09
LiveCC-Instruct	1.47	1.07	1.01	1.07	1.12	1.08	2.54	1.23	1.02	2.66	1.97	1.25
Real-time proactive												
MageVL	1.79	1.08	1.05	1.11	1.32	1.06	1.49	1.20	1.05	1.00	1.53	1.08
JoyAI-VL	**4.89**	4.18	1.43	**3.51**	**3.66**	**2.78**	<u>4.89</u>	4.20	1.39	**3.72**	<u>3.83</u>	<u>2.62</u>
Proact-VL	<u>4.88</u>	<u>4.25</u>	**2.34**	3.39	3.21	<u>2.60</u>	**4.90**	4.20	<u>2.12</u>	3.48	3.75	**2.71**
ONSTEP	4.85	**4.39**	<u>2.20</u>	<u>3.47</u>	3.32	2.55	**4.90**	**4.34**	**2.24**	3.34	**3.87**	2.58

E.2 Is the LLM judge robust?

Three of our six delivery metrics and the PAUC score are produced by an LLM judge, so a natural objection is that the reported ordering reflects the judge's idiosyncrasies rather than a property of the systems. We test this by re-scoring both our method and the strongest baseline, Proact-VL, with a second judge(Claude Sonnet-5) in place of GPT-5.6-Luna on the main set, holding the predictions, the rubric and the reference fixed.

ONSTEP is ahead of Proact-VL on 9 of the 10 judged quantities under both judges. The per-dimension scores of the two judges correlate at $r = 0.9996$ for ONSTEP and $r = 0.9997$ for Proact-VL, so the judges agree almost perfectly on which dimensions are strong and which are weak.

Sonnet-5 is systematically harsher on the 1-5 delivery dimensions, and the effect is concentrated in FinalQ: Completeness alone falls by 1.12 and 1.31 points. In the opposite direction it is system-

Table 11: **Judge robustness.** Identical predictions scored by two judges. pAUC is reported at $\omega = 0.5$; LiveU and FinalQ are 1–5; AnswerQ is the mean over answer dimensions. Δ is ONSTEP − Proact-VL.

Judge	Method	pAUC	LiveU			FinalQ			AnswerQ
			Time	Rate	TextU	Verd.	Cont.	Compl.	Match
Luna	Proact-VL	37.3	3.23	2.81	4.47	2.84	2.78	4.36	3.44
	ONSTEP	**38.0**	**3.50**	**3.87**	**4.74**	**3.12**	**3.33**	**4.39**	**3.46**
Sonnet-5	Proact-VL	39.9	3.10	2.77	4.02	2.28	2.19	3.05	3.48
	ONSTEP	**41.5**	**3.41**	**3.44**	**4.43**	**2.31**	**2.57**	**3.27**	3.48
Δ	Luna	+0.7	+0.27	+1.06	+0.27	+0.28	+0.55	+0.03	+0.02
	Sonnet-5	+1.6	+0.31	+0.67	+0.41	+0.03	+0.38	+0.22	0.00

atically more generous on PAUC. Because the shift is close to common across systems it largely cancels in the comparison.

However, agreement on the sign of the gap does not imply agreement on its size: the correlation between the two judges' per-dimension *margins* is only $r = 0.62$, far below the correlation of the raw scores. Two dimensions move substantially. Our Rate advantage shrinks from $+1.06$ to $+0.67$, and our Verdict advantage all but disappears, from $+0.28$ to $+0.03$, under Sonnet-5 the two systems are indistinguishable on verdict correctness. Conversely the PAUC margin grows under Sonnet-5. The single sign flip is AnswerQ, where Sonnet-5 places Proact-VL ahead by 0.002; at that magnitude the two systems are tied, and both judges agree that answering ability is where the two methods differ least.

We therefore read the result as follows: claims about *ordering*, that ONSTEP delivers guidance better paced, clearer and better timed than the strongest baseline, survive a change of judge.

F Sensitivity to the speaking threshold

The onset head emits a per-second probability of a speaking event and the LLM-decoder speaks when it exceeds a threshold τ. Since τ is not learned, a natural concern is that the reported results depend on having tuned it. We sweep τ from 0.10 to 0.99 on the main split, holding the checkpoint, the decoder and the reference fixed.

Table 12: **Threshold sweep** on the main set.$\tau = 0.30$ is the value throughout the paper.

τ	F1 ↑	P ↑	R ↑	TimeDiff ↓
0.10	36.92	32.98	**41.93**	3.62
0.20	39.28	36.95	**41.94**	**3.38**
0.30	39.70	38.90	41.10	3.43
0.40	39.76	40.02	39.51	3.53
0.50	**40.20**	42.01	38.53	3.54
0.60	39.89	42.93	37.25	3.68
0.70	40.15	44.74	36.41	3.73
0.80	39.56	46.69	34.33	4.05
0.90	39.75	51.80	32.25	4.38
0.99	40.02	**68.29**	28.31	5.20

Precision rises monotonically from 32.98 to 68.29 (a factor of 2.07) and recall falls from 41.93 to 28.31 across the sweep. Every point is ordered; there is no region in which the head behaves erratically. The two move almost exactly in compensation, so F1 is nearly invariant: over $\tau \in [0.30, 0.99]$ it spans 39.56-40.20, a range of 0.64 with a standard deviation of 0.20, so across two thirds of the probability range the threshold is not a distinguishable factor at all. Only at $\tau = 0.10$ does F1 drop meaningfully (36.92), where the head speaks so freely that precision falls faster than recall rises.

Because F1 is insensitive, it is the wrong instrument for choosing τ; the operating point is visible instead in the timediff column, which has an interior minimum at $\tau = 0.20$ (3.38) and degrades in both directions, to 3.62 at $\tau = 0.10$ and monotonically to 5.20 at $\tau = 0.99$. The optimum is therefore bracketed rather than sitting at the edge of the swept range.

The coverage normalisation is what makes this column informative. *Raw* TimeDiff, averaged only over utterances the model answered, in fact improves monotonically as τ rises, from 1.98 s to 1.70 s: a stricter gate fires only on confident onsets, so the responses that do occur are prompter. Taken alone that would recommend $\tau = 0.99$, which answers barely a third of the reference utterances. Dividing by coverage prices the ignored utterances back in and reverses the conclusion, which is why we never report raw TimeDiff without it.

Choice of $\tau = 0.30$. We use $\tau = 0.30$ throughout. It is within 0.35 F1 of the best value in the sweep, and within 0.06 of the best TimeDiff score, whereas the extremes optimise one at the clear expense of the other ($\tau = 0.99$ reaches precision 68.3 but answers barely a third of reference utterances). We stress that this is a balance rather than a tuned optimum: the results in Tables 1 and 2 would be substantively unchanged anywhere in $[0.20, 0.70]$.

G Does the step bank need procedural context?

Equation 7 represents each recipe step using its causally encoded plan prefix rather than encoding the step independently. We ablate this choice by replacing the causal-prefix bank with isolated step embeddings while holding the remaining training configuration fixed.

One failure mode of isolated encoding is repeated instructions. Our dataset contains 81 within-video pairs with identical step text. For example, *"Now clean the knife by wiping it with a paper towel"* occurs twice in one video. An isolated encoder receives identical input for both occurrences and therefore produces identical representations with cosine similarity 1.000. The step objective cannot distinguish these occurrences regardless of the encoder layer or learned projection. Causal-prefix encoding breaks this symmetry because each occurrence is conditioned on a different preceding recipe history.

Table 13: **Effect of step-bank construction** on the Main Set. All configurations include the onset head and rate regularizer and differ only in their use and construction of the step bank.

Step bank	F1 ↑	TimeDiff ↓	PAUC ↑
No step objective	38.19	3.75	37.80
Isolated steps	37.21	3.84	37.80
Causal prefix (ONSTEP)	**39.70**	**3.43**	**38.03**

Table 13 shows that the isolated step objective performs worse than using no step objective, decreasing F1 by 0.98 points and increasing TimeDiff by 0.09 seconds. In contrast, causal-prefix representations improve F1 by 1.51 points, reduce TimeDiff by 0.32 seconds, and improve PAUC by 0.23 points relative to using no step objective.

Why identical targets are problematic. Suppose two steps i and j have identical isolated representations, such that $k_{v,i} = k_{v,j}$. For a query whose target is i, the two steps receive identical logits. Consequently, the probability assigned to the correct step is bounded by

$$p(c_t = i \mid q_t) \leq \frac{\exp(\gamma q_t^\top k_{v,i})}{2\exp(\gamma q_t^\top k_{v,i})} = \frac{1}{2}, \tag{16}$$

and the corresponding InfoNCE loss is at least $\log 2$. More importantly, both occurrences provide the same step-level supervision even though they represent different points in the procedure. The isolated objective therefore cannot supply the positional distinction it was introduced to learn. Encoding the recipe prefix gives repeated instructions distinct, order-aware targets and turns the step objective into a useful downstream training signal.

G.1 STEP-ANCHORED OBJECTIVE DETAILS

Order-aware step bank. We construct the step bank in eq. (7) using the frozen text decoder of Qwen3-VL. Steps are concatenated in recipe order with explicit delimiters, and the final-token hidden state is extracted after each prefix. Thus, $z_{v,s}$ depends only on steps $1, \ldots, s$. This causal construction encodes both the current instruction and its procedural history. In particular, two identical instructions at different recipe positions receive different representations because their preceding contexts differ; encoding each instruction independently would instead produce identical targets.

Projection and similarity. Let g_t denote the hidden state of <|STEP|> at streaming layer ℓ_s. Because g_t and $z_{v,s}$ may belong to different representation spaces, we map them into a shared embedding space using trainable projections W_q and W_k:

$$q_t = \frac{W_q g_t}{\|W_q g_t\|_2}, \qquad k_{v,s} = \frac{W_k z_{v,s}}{\|W_k z_{v,s}\|_2}. \tag{17}$$

The normalized inner product $q_t^\top k_{v,s}$ therefore measures cosine similarity between the streaming state and candidate step s.

Optimization. The within-video objective in eq. (8) contrasts the active step against every other step in the same recipe. These negatives are challenging because they often describe semantically related actions and may contain repeated objects or verbs. Consequently, minimizing the loss requires the streaming representation to capture procedural position rather than only coarse semantic similarity. The step encoder and its representations remain frozen, while W_q, W_k, the learned logit scale γ, and the streaming backbone are optimized jointly. Since <|STEP|> occurs before <|FLAG|> in the token sequence, this supervision can shape the representation subsequently used by both the onset gate and the response decoder.

H QUALITATIVE COMPARISON

Figures 7, 8, 9, 10, 11, and 12 place both methods on one axis: ONSTEP on the top track, the Proact-VL baseline below it, the ground truth annotation third, and the user's spoken questions last. Because the tracks share a time axis and a colour scheme, the comparison is visual, a bar sitting above an annotation bar of the same colour is a hit in category and in time, and a bar with nothing beneath it is output the session never called for. Two differences are visible on every slide. The baseline speaks too much and too late: it exceeds the annotated utterance count on four of the six figures, by as much as 77 utterances against 40 annotated on the longest, and its output sits a median of 2–14 s from the nearest annotated event against 1–6 s for ours. It also restarts within five seconds of its own previous utterance two to three times as often (6–26 times per session against 2–14), which reads as clusters of bars where the annotation has one. ONSTEP tracks the annotation more closely on both counts, and is ahead on F1 ($+0.7$ to $+13.5$), precision ($+2.4$ to $+14.3$) and combined TimeDiff on all six. The margin grows with video duration, the three videos over 750 s give the three largest precision gaps; the longest session also shows the limit of ours, which degrades in the same direction, only later.

H.1 FAILURE CASES.

The examples in figs. 13 and 14 reveal two complementary limitations. In the first case, the model responds at the correct time but reports eight remaining steps when the current instruction is in fact the final step. This is a recurring error: 83 of 131 step-count questions receive an incorrect count. Over long videos, the bounded KV cache may evict earlier context, making it difficult to recover the exact current position and total number of steps. Although the auxiliary step objective encourages position-aware representations, inference does not explicitly route step-count questions through the <|STEP|> representation. Such routing could improve performance, but would introduce question-type-specific inference logic rather than preserving a unified streaming policy. The second case instead reflects a perceptual failure: the model speaks within the correct temporal window but describes an incorrectly executed action as successful. Determining whether the jelly was scooped or spread properly requires fine-grained temporal perception and comparison against the intended

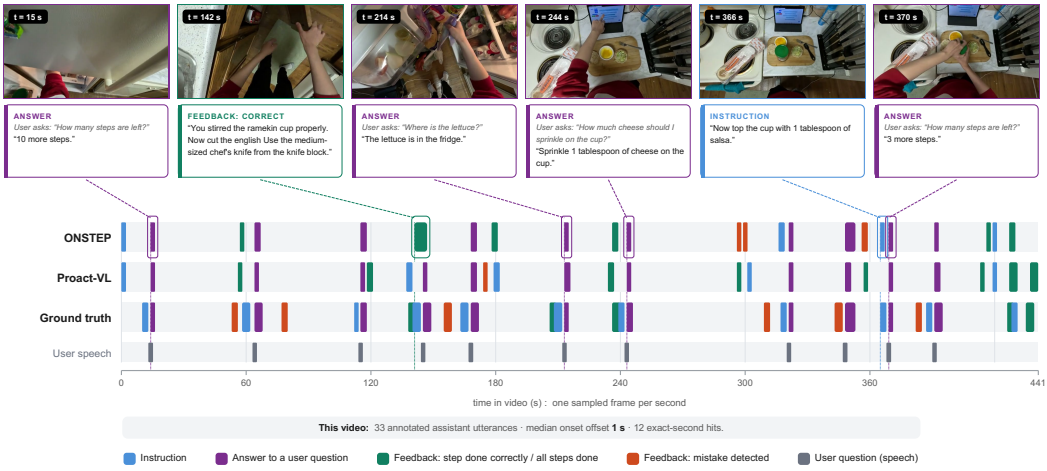


Figure 7: microwave egg sandwich

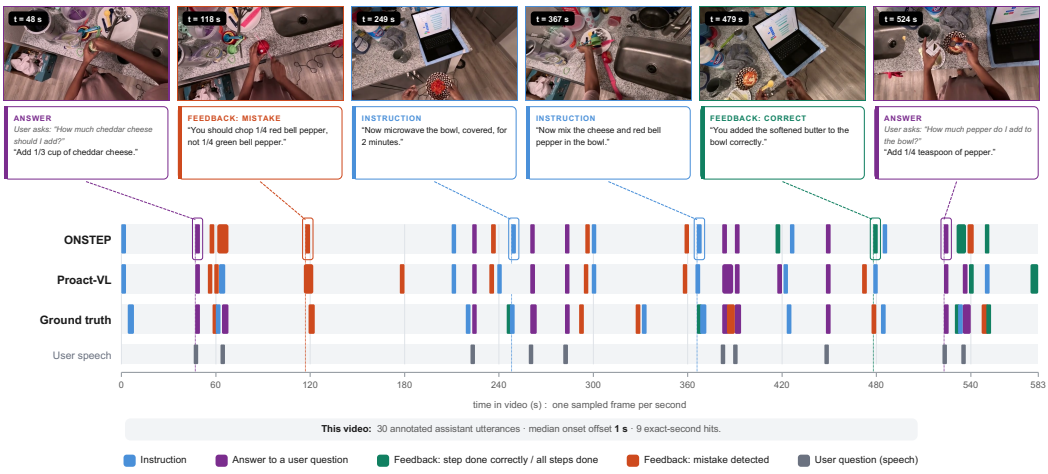


Figure 8: microwave cheese and pepper bowl

procedure, rather than simply recognizing the objects and action involved. This remains challenging for current multimodal language models, which can identify coarse actions while overlooking subtle execution errors or defaulting to the more likely assumption of successful completion. Indeed, 87 of 438 reference mistakes are incorrectly answered with positive feedback. Together, these cases show that accurate response timing does not guarantee accurate content: failures can arise either from limited long-term procedural memory or from insufficiently fine-grained visual verification.

Limitations. The question-answer pairs are human-verified, they are initially generated automatically and may not fully reflect natural user interactions. The model also struggles with long-horizon step tracking and subtle visual mistakes. Finally, several evaluation metrics rely on LLM judges, which may introduce evaluation bias despite our robustness analysis.

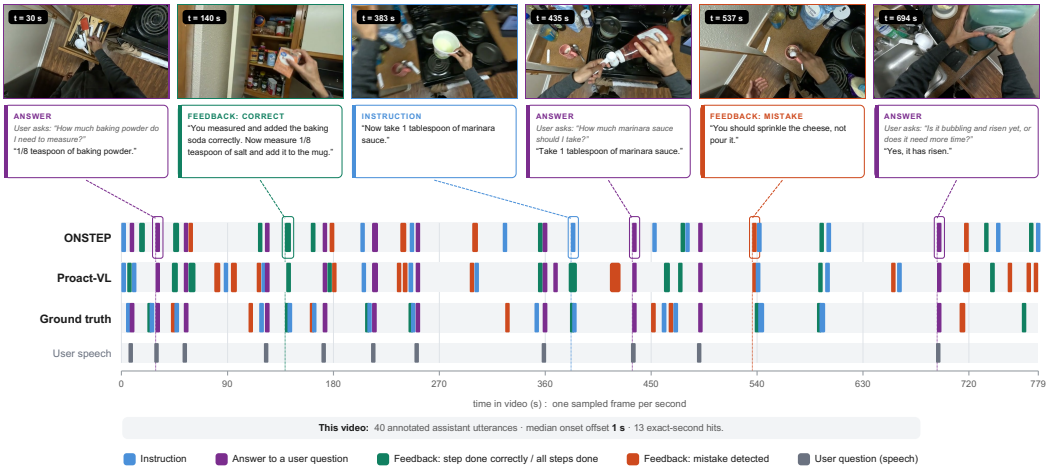


Figure 9: microwave mug pizza

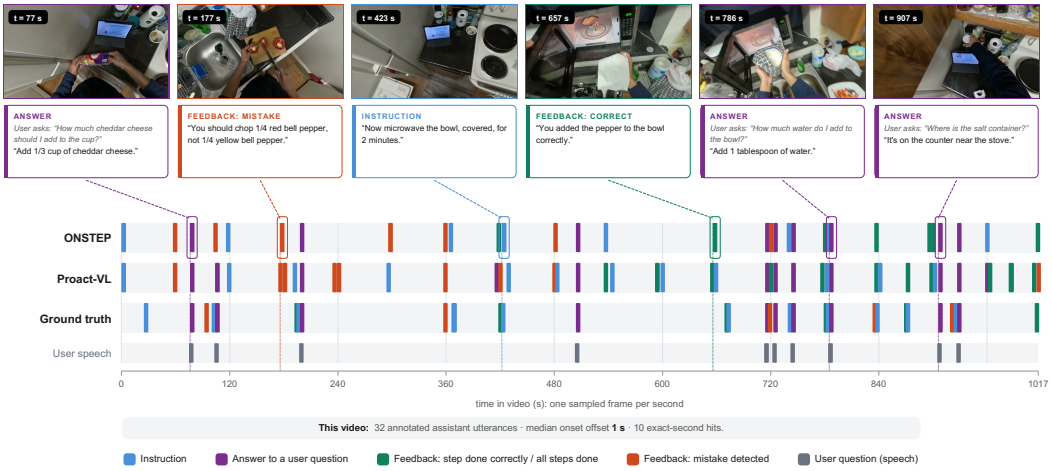


Figure 10: microwave cheese and pepper bowl

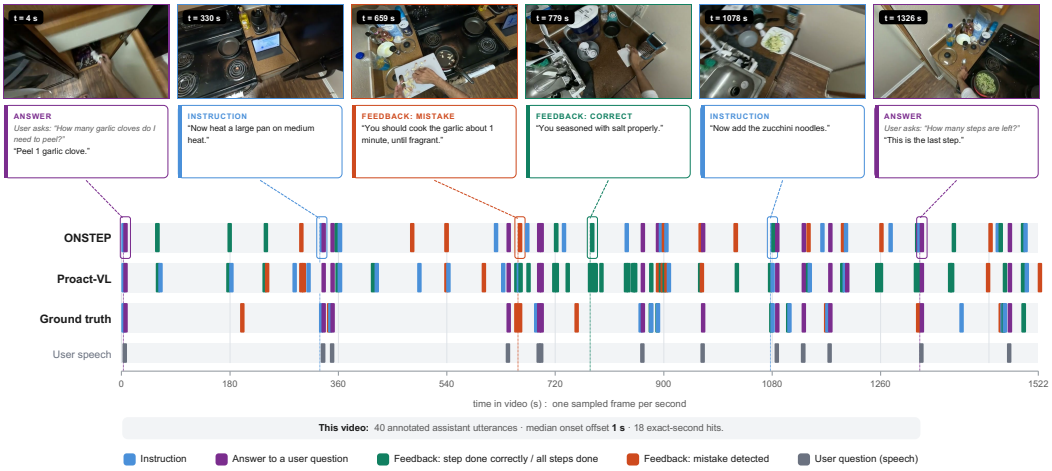


Figure 11: garlic butter pasta

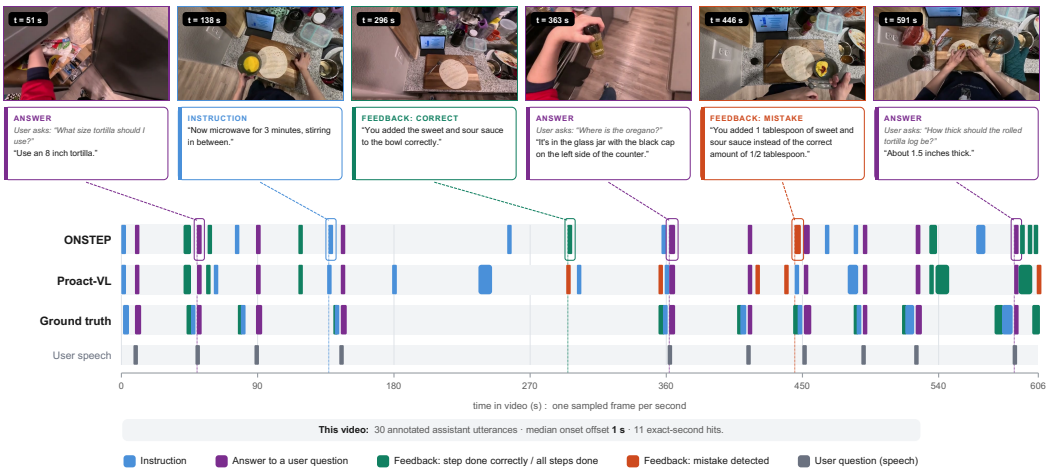


Figure 12: egg and tortilla wrap

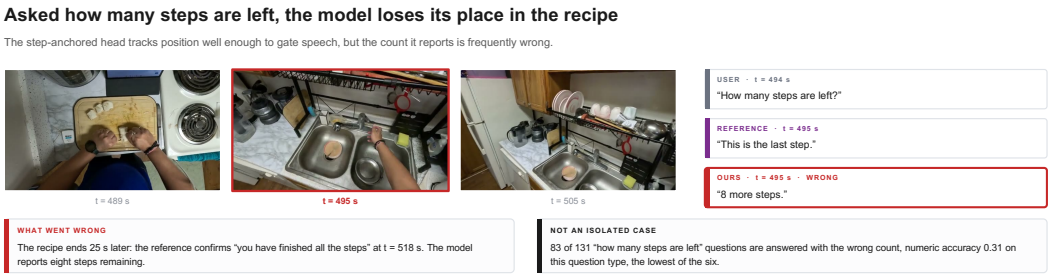


Figure 13: wrong step-count

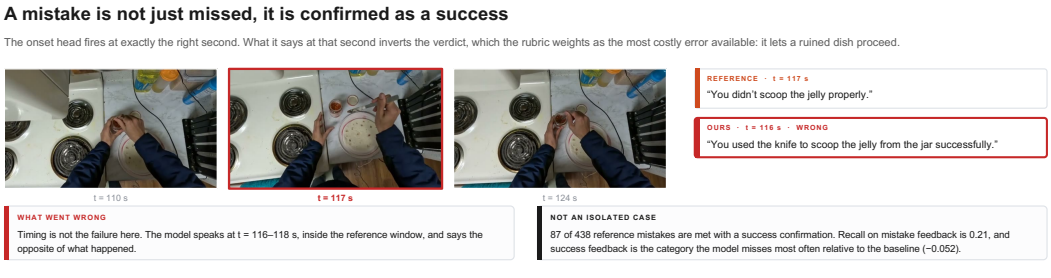


Figure 14: perception error