STRUCTURE-OPTIONAL LEARNING FOR PROACTIVE ASSISTANCE

Anonymous authors
Paper under double-blind review

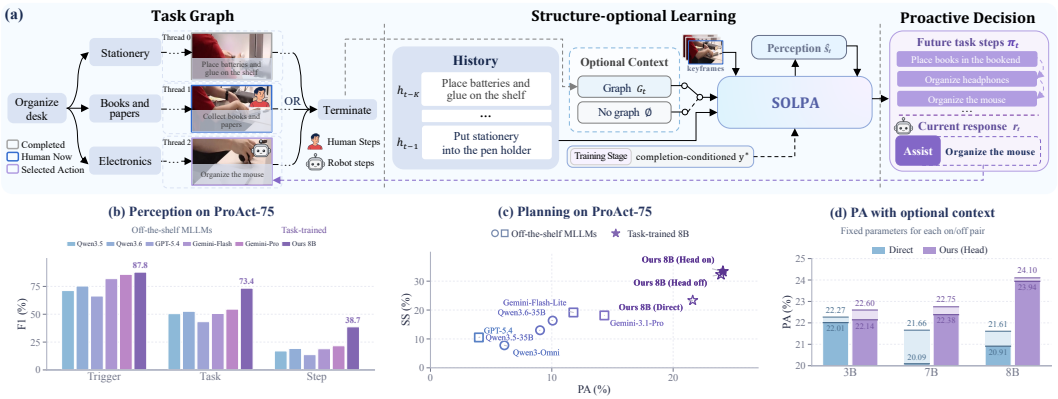


Figure 1: **Structure-optional proactive assistance on ProAct-75.** Structure guides training but remains optional at inference. (a) An illustrative two-stage example combines video perception, completed history, and optional graph context to select a parallel action. (b) Perception F1 covers trigger, task, and step recognition. (c) Our task-trained 8B and off-the-shelf MLLMs are compared in SS and PA under their respective input settings. (d) Our HEAD exceeds same-backbone Direct in graph-off PA for all three models. Bar tops mark graph-on PA and internal lines mark graph-off PA.

ABSTRACT

Proactive assistants advance shared tasks without waiting for explicit requests. Their actions must respect task prerequisites and allow parallel progress alongside a person. Task graphs encode these prerequisites and parallel branches but may be unavailable at inference. We introduce SOLPA (Structure-Optional Learning for Proactive Assistance), which trains a Planner using task-graph supervision. Completion-conditioned questions assume an unfinished step is complete while keeping the person's current step fixed. Graph-derived answers supervise its next assistance action conditioned on completed-step history. Randomly omitting graph text during training lets the same parameters support inference with and without graphs. On ProAct-75 with predicted task state and annotated completed-step history, our 8B system achieves 33.37% step-success (SS) and 24.10% parallel-action rate (PA), improving over same-backbone Direct by 10.01 and 2.49 percentage points, respectively. Under the same scorer, the newly evaluated strong closed-source models reach at most 14.33% PA without task-specific training. With parameters fixed, the Planner retains 23.94% PA without graph text or its action-name constraint. Our code is available at `https://github.com/only4anonymous/SOLPA.git`.

1 INTRODUCTION

Proactive assistance advances a higher-level goal without waiting for an explicit request, requiring continuous observation to decide when to act and what to do (Patel & Chernova, 2022; van Den Broek & Moeslund, 2024; Lu et al., 2025). For an assistant working alongside a person, anticipating the

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.

person's next step does not by itself identify work that can proceed in parallel (Zhu et al., 2026). For example, an assistant can prepare a fresh bag while the person removes the used one from a bin. Delegating task allocation to a robot can reduce perceived workload (Karakikes & Nathanael, 2023), while greater autonomy can reduce the need for human instructions (Schlafly et al., 2024). We study this setting through video-based proactive assistance, in which an assistant monitors a person's activity and decides when to intervene and which action to take.

Selecting an action requires reasoning about the person's intentions (Zhu et al., 2025), recognizing the current task state, and considering which actions are feasible in the current scene (Brohan et al., 2023). It also requires knowledge of task structure: the steps, their prerequisites, and independent branches encoded by a task graph (Kaelbling & Lozano-Pérez, 2011; Gombolay et al., 2018). Prior work uses this knowledge to derive training targets for video representations (Zhou et al., 2023), guide differentiable planner decoding (Seminara et al., 2026), or supply retrieved plan recommendations as context (Nagasinghe et al., 2024). For proactive assistance, ProAct-Helper selects actions through external graph-guided search at inference (Zhu et al., 2026). We study how graph-guided action preferences can supervise a Planner offline, enabling the model itself to select assistance actions while keeping graph input optional at inference.

Completed work can change which feasible action the assistant should select, making completed-step history a key planning input. Completion-conditioned question answering (QA) uses task graphs to supervise action reselection after hypothetical step completion while holding the person's current step fixed. Graph-context dropout complements this supervision by training the same Planner on ordinary planning examples with graph text omitted.

To this end, we introduce SOLPA (Structure-Optional Learning for Proactive Assistance), a structure-supervised approach to proactive planning (seen in Figure 1). Within ProAct-Helper's perception and planning framework, our Planner selects the assistant's next action using the current state and completed-step history. We use ProAct-Helper's graph-based action preference offline to recompute the recommended parallel action after marking a candidate step complete, while keeping the person's current step fixed (Figure 3). We pair this supervision with graph-context dropout, which omits graph inputs from a fraction of planning examples without changing their targets. The same Planner supports graph-on inference with graph text and action names restricted to the graph's vocabulary, and graph-off inference without either.

Our contributions are summarized as follows:

- We introduce SOLPA to learn proactive action selection from task-graph supervision with optional graph input at inference. With fixed Planner parameters, paired 3B/7B/8B evaluations yield 22.14–23.94% parallel-action rate (PA) without graph text or its action-name constraint.
- Completion-conditioned QA provides supervision for action reselection after hypothetical progress. A controlled 3B ablation with annotated task state (oracle perception) shows PA gains of 3.84 and 4.94 points over training with history and graph context in graph-on and graph-off modes. It also improves both metrics over current-state QA with matched data budgets and training settings.
- System-level comparisons on ProAct-75 show stronger step recognition and model-decoded planning than same-backbone Direct, which jointly predicts state and actions from video, across three backbone configurations. In annotated-state simulation, the 8B model mirrors the person's next step on 2.8% of simulation time steps compared with 11.9% for Direct, while saving task steps without repeating that next step on 34.9% compared with 20.7%.

2 Related Work

Proactive assistance and procedural understanding. Proactive assistants use observations to anticipate opportunities to help (Koppula & Saxena, 2015; Patel & Chernova, 2022; van Den Broek & Moeslund, 2024; Lu et al., 2025). This setting connects intention inference (Rabinowitz et al., 2018; Zhu et al., 2025) with procedural state recognition. COIN, CrossTask, Ego-Exo4D, and Assembly101 provide task and step annotations (Tang et al., 2019; Zhukov et al., 2019; Grauman et al., 2024; Sener et al., 2022). Paprika uses procedural knowledge to supervise video representations (Zhou et al., 2023). ViterbiPlanNet injects a procedural knowledge graph into planner decoding (Seminara et al., 2026), and KEPP retrieves plans from a graph of step transitions (Nagasinghe et al., 2024). ProAct-75 provides AND/OR task graphs, and ProAct-Helper uses graph-guided action selection at

inference (Zhu et al., 2026). We reuse its action preference offline to supervise completion-conditioned choices that the Planner learns to decode directly.

Learning from privileged context. Privileged information available during training can guide learning without being required at inference (Vapnik & Vashist, 2009; Kamienny et al., 2020). Context distillation transfers behavior induced by richer context to models receiving reduced context (Snell et al., 2022), and recent methods jointly learn policies with and without privileged information using shared parameters (Penaloza et al., 2026). We study procedural supervision for proactive action selection. Our auxiliary action targets are derived from task graphs after hypothetical completion updates while holding the human's current step fixed. These targets supervise a Planner trained with and without explicit graph context.

Structured planning and constrained generation. Explicit task representations support task and motion planning (Kaelbling & Lozano-Pérez, 2011), and affordance constraints ground language-based action selection (Brohan et al., 2023). Neuro-symbolic methods combine language models with explicit action reasoning (Wu & Liu, 2025). Constrained decoding can restrict generation to predefined names (De Cao et al., 2020). Inspired by dropout (Srivastava et al., 2014; Kamienny et al., 2020), we omit complete graph text from some planning examples while retaining completed-step history and completion-conditioned supervision. The Planner directly selects actions in both graph modes.

3 METHOD

SOLPA separates perception from planning (Figure 2). After defining their inputs and outputs, we describe history-conditioned planning (Section 3.2), optional graph context (Section 3.3), and QA about hypothetical step completion (Section 3.4). Section 3.5 combines the training objectives.

3.1 PROBLEM FORMULATION

ProAct-75 (Zhu et al., 2026) evaluates five vision-based tasks for proactive response. **Trigger detection** determines whether a response is needed, while **task detection** and **step detection** identify the ongoing task and the person's current step. **Future action prediction** predicts a future action sequence, and **proactive action selection** chooses the assistant's next graph-feasible action or WAIT.

At time t, $s_t = (p_t, z_t^{\text{task}}, z_t^{\text{step}})$ denotes the binary response indicator, current task, and human step. A Perceiver with parameters w_p estimates $\hat{s}_t$ from video $V_{\leq t}$ observed up to t. The Planner takes $s_t = \hat{s}_t$ as input. Oracle-perception evaluation supplies the annotated state $s_t = s_t^*$ to isolate planning from perception errors. If $p_t = 0$, the system returns WAIT without invoking the Planner. Otherwise, the Planner predicts an ordered future task-step sequence π_t, followed by r_t, its next action or WAIT.

Each task has an annotated graph G with AND/OR prerequisites and parallel branches. Execution threads distinguish separate branches, including parallel sub-actions within one task category. Let D_t denote all completed steps and G_t the local text view of G constructed from D_t and the current step. The Planner sees the latest K completed steps as ordered history $H_{<t}^{(K)}$. Its optional graph input is $C_t \in \{G_t, \varnothing\}$. The Planner Q_{w_d}, with parameters w_d, predicts for $p_t = 1$ as follows:

$$(\pi_t, r_t) = Q_{w_d}(V_{\leq t}, s_t, H_{<t}^{(K)}, C_t). \tag{1}$$

The graph-on setting supplies $C_t = G_t$ and restricts action names to the full task graph's vocabulary. This vocabulary constraint does not check whether current prerequisites are satisfied. The graph-off setting uses $C_t = \varnothing$ and unconstrained decoding. Both modes share w_p and w_d.

3.2 HISTORY-CONDITIONED PLANNING

A frame may reveal the current human step without showing progress on other branches. The Planner uses $H_{<t}^{(K)}$ to condition future-step prediction and assistant-action selection in both graph modes.

The training output lists future steps π_t^* before the assistant action r_t^*, where $*$ marks a target. ProAct-75's graph manager, a rule-based program that checks prerequisites and tracks execution threads,

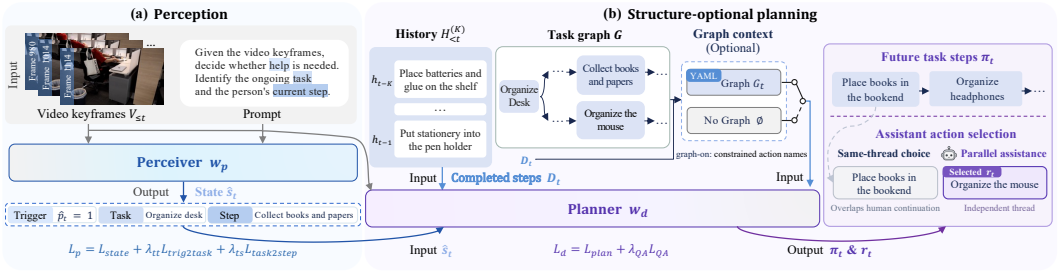


Figure 2: **Overview of SOLPA.** The system estimates task state before choosing an assistant response. (a) The Perceiver predicts $\hat{s}_t$ from video, and $\mathcal{L}_p$ combines state prediction with hierarchical binding. (b) The Planner uses video, state, completed history, and optional graph text to predict future task steps π_t and assistant response r_t, while $\mathcal{L}_d$ combines planning with completion-conditioned QA. Both graph modes share parameters, with graph-on constraining action names to the full task graph's action vocabulary. Illustrative choices contrast overlap with the person's anticipated next step and assistance on another thread, using shortened labels for display.

supports offline target generation (Zhu et al., 2026). The sequence follows graph-ordered remaining steps or falls back to annotated future steps. The remaining-step format ends in "Terminate". Graph-based rules provide a feasible assistant action or WAIT. These targets remain unchanged when graph context or QA is omitted. Target construction and fallbacks are detailed in Section A.2.

During training, annotated state fields are supplied as an input prefix, giving $x_t^* = (V_{\leq t}, s_t^*, H_{<t}^{(K)}, C_t)$. Let $\ell_w(y \mid x)$ denote mean negative log-likelihood over target tokens. Only future-step and action tokens contribute to the planning loss, defined as follows:

$$\mathcal{L}_{\text{plan}} = \ell_{w_d}((\pi_t^*, r_t^*) \mid x_t^*). \tag{2}$$

At inference, generating π_t before r_t lets action selection use the predicted continuation. Prompts and output examples for both stages appear in Section E.

3.3 TASK STRUCTURE AS OPTIONAL CONTEXT

Completed history records progress, while task graphs describe dependencies and independent branches (Kaelbling & Lozano-Pérez, 2011; Nagasinghe et al., 2024; Seminara et al., 2026). We train the Planner with optional text describing this structure.

We extract G_t around the current step, retaining connecting completed and intermediate nodes. Its local frontier lists visible next actions whose prerequisites would hold after the current step. Other feasible actions can lie outside this view. Compact YAML records names, state tags, edges, and frontier information. The full graph supplies AND/OR conditions for feasibility checks, while the text shows local connectivity. Serialization details and representation diagnostics appear in Sections A.1 and C.5.

For trigger-positive planning examples with usable graph text, we omit G_t with probability d and retain the same targets. The optional input is sampled as follows:

$$C_t = \begin{cases} G_t, & \text{with probability } 1-d, \\ \varnothing, & \text{with probability } d. \end{cases} \tag{3}$$

Graph-context dropout trains shared Planner parameters while retaining video, state, and history.

Graph-on also applies vocabulary-constrained decoding (De Cao et al., 2020), detailed in Section A.1. The main graph-off setting removes both graph text and this action-name constraint.

3.4 LEARNING FROM HYPOTHETICAL STEP COMPLETION

Ordinary planning examples supervise action selection at observed states. We add *completion-conditioned QA* to supervise selection after hypothetical progress. Each question assumes one unfinished step is complete while keeping the person's current step fixed. It asks for a cross-thread action, on an execution thread different from the person's current thread.

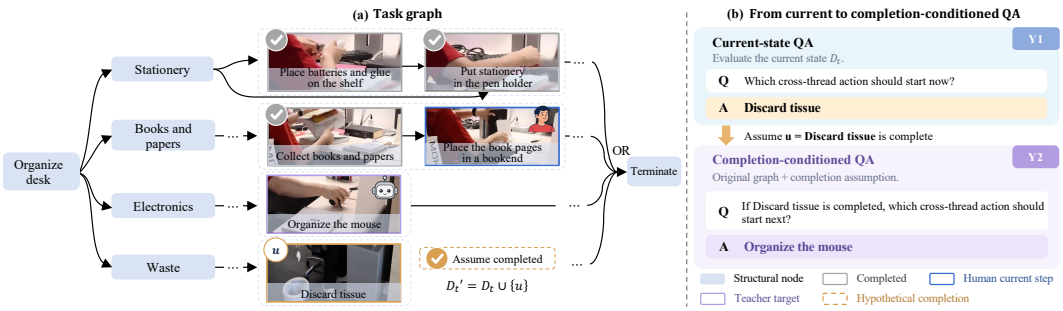


Figure 3: **Completion-conditioned action selection in Organize Desk.** Graph-based labels illustrate how hypothetical completion changes assistance. (a) Completed steps and the current bookend step define the state. The tissue photo is from another recording. (b) Current-state QA (Y1) recommends "Discard tissue", whereas completion-conditioned QA (Y2) recommends the already-eligible "Organize the mouse" after hypothetical tissue disposal, holding the human step fixed.

Constructing questions and answers. We select an unfinished step u other than the person's current step, prioritizing feasible cross-thread actions. We form $D'_t = D_t \cup \{u\}$ while holding the person's current step fixed. The question $q_t(u)$ states this assumption and asks for the next action or WAIT. An offline rule-based selector A_G uses the full graph to define the target as follows:

$$y^*_{t,u} = A_G(D'_t, z_t^{\text{step}}). \tag{4}$$

A_G selects a feasible cross-thread action with the lowest ProAct-Helper thread-mixing entropy (Zhu et al., 2026), or returns WAIT if none exists. This rule favors human and assistant work on different threads. In Figure 3, assuming "Discard tissue" is complete changes the recommendation to the already-eligible "Organize the mouse". Thus, completion can change the preferred action without unlocking a new one. Section A.2 details candidate selection, history initialization, and tie-breaking.

Training the Planner with QA. The text-only QA input consists of $q_t(u)$ and G_t^{QA}, the local graph text of the *original* state. The question conveys the hypothetical completion, while A_G computes the answer from the updated set D'_t. The Planner generates the answer using the same w_d as ordinary planning. The QA loss is defined as follows:

$$\mathcal{L}_{\text{QA}} = \ell_{w_d}(y^*_{t,u} \mid q_t(u), G_t^{\text{QA}}). \tag{5}$$

At inference, the Planner generates actions without A_G or auxiliary questions.

3.5 TRAINING OBJECTIVES

We optimize the Perceiver and Planner separately. Annotated trigger, task, and step fields supervise $\mathcal{L}_{\text{state}} = \ell_{w_p}(s^*_t \mid V_{\le t})$. The Perceiver combines this loss with ProAct-Helper's hierarchical binding (Zhu et al., 2026) as follows:

$$\mathcal{L}_p = \mathcal{L}_{\text{state}} + \lambda_{\text{tt}}\mathcal{L}_{\text{trig2task}} + \lambda_{\text{ts}}\mathcal{L}_{\text{task2step}}. \tag{6}$$

Binding aligns projected trigger with task and task with step representations (Section A.3). Each planning minibatch is paired with a QA batch, giving the Planner objective as follows:

$$\mathcal{L}_d = \mathcal{L}_{\text{plan}} + \lambda_{\text{QA}}\mathcal{L}_{\text{QA}}. \tag{7}$$

Both losses update w_d after separate averaging, with QA weighted by λ_{QA} and omitted in the no-QA ablation. Graph-context dropout applies only to planning examples. QA retains graph text. Loss and sampling details appear in Sections A.3 and B.

4 EXPERIMENTS

4.1 EVALUATION SETUP

Benchmark. We evaluate on ProAct-75 (Zhu et al., 2026), which provides 4,074 training and 1,309 test videos. Unless otherwise stated, we use its best-view split with 1,905 training and 516

Table 1: **Main results on ProAct-75.** Shaded and †-marked rows are evaluated here using model-decoded actions. Other scores are published references, with ProAct-Helper‡ SS/PA using an external graph-guided selector. PA denominators are effective non-Wait actions for published scores and all scored decisions for newly evaluated scores. The two-stage HEAD uses predicted state and annotated completed history. Bold/underline mark the best/second-best shaded scores.

	Trigger (%)				Task (%)				Step (%)						
Base Model	mAcc	mF1	Acc	F1	mAcc	mF1	Acc	F1	mAcc	mF1	Acc	F1	ED↓	SS (%)↑	PA (%)↑
Open-Source MLLMs															
Qwen3-VL-30B-A3B-Instruct	51.61	66.25	73.50	81.89	22.80	31.70	34.66	51.48	6.67	9.44	8.53	15.72	4.77	1.30	0.09
Qwen3-Omni-30B	46.86	63.35	65.07	71.30	18.96	26.87	17.55	29.87	3.72	5.42	4.48	8.58	4.64	7.80	6.09
Qwen2.5-VL-32B-Instruct	45.72	62.36	63.77	69.64	16.62	23.96	29.45	45.50	4.04	6.04	6.14	11.57	4.72	3.80	3.71
Qwen2.5-VL-3B-Instruct	42.33	54.94	69.55	80.59	12.53	18.92	23.18	37.64	1.46	2.42	3.06	5.95	4.74	3.40	4.23
Qwen3.5-35B-A3B†	44.23	60.51	63.44	71.26	11.14	16.03	33.80	50.53	2.89	4.82	9.31	17.04	4.72	13.01	9.03
Qwen3.6-35B-A3B†	47.81	63.70	67.44	75.35	12.03	17.24	35.74	52.65	3.90	6.32	10.67	19.28	4.71	16.34	10.07
Closed-Source MLLMs															
Qwen3-VL-Flash	54.73	70.35	72.06	77.48	17.36	24.84	17.91	30.37	3.70	5.51	2.94	5.72	4.76	4.60	2.46
Qwen3-VL-Plus	51.52	67.39	69.89	76.41	14.88	21.52	17.39	29.63	5.14	7.38	4.66	8.91	4.71	4.50	3.10
GPT-4o	30.59	46.70	47.11	51.38	16.67	24.90	20.27	33.71	4.02	6.05	8.70	16.00	4.64	2.60	0.77
GPT-5.4†	41.52	58.21	59.81	66.37	10.21	14.91	27.65	43.32	2.17	3.58	7.37	13.72	4.74	10.52	3.99
Gemini-2.5-Flash	49.73	64.24	72.52	81.44	21.78	30.66	31.14	47.49	6.97	10.12	9.42	17.21	4.70	6.20	2.55
Gemini-2.5-Pro	42.89	55.94	69.31	80.21	22.95	32.39	35.23	52.11	8.69	12.31	11.99	21.41	4.70	12.00	3.83
Gemini-3.1-Flash-Lite†	45.16	58.25	71.73	81.97	12.14	17.31	33.94	50.68	3.42	5.62	10.51	19.02	4.75	19.16	11.78
Gemini-3.1-Pro-Preview†	47.94	60.05	76.70	85.84	14.75	20.90	37.54	54.59	5.27	8.51	12.16	21.68	4.72	18.14	14.33
Based on Qwen2.5-VL-3B-Instruct															
ProAct-Helper‡	61.50	75.38	79.32	85.23	28.33	36.72	51.03	67.58	7.61	10.67	18.68	31.49	3.96	36.60‡	17.44‡
SOLPA (Direct)	53.00	66.76	76.87	85.09	20.82	27.37	54.82	70.81	2.99	4.67	16.93	28.96	4.00	28.00	22.27
SOLPA (HEAD)	64.68	77.94	81.33	86.59	36.62	46.79	53.50	69.70	8.94	12.42	20.14	33.53	3.85	30.37	22.60
Based on Qwen2.5-VL-7B-Instruct															
ProAct-Helper‡	62.90	76.56	80.08	85.65	27.07	34.79	52.91	69.20	8.25	11.56	19.85	33.13	3.90	36.10‡	19.41‡
SOLPA (Direct)	64.29	77.56	81.42	86.87	19.80	26.11	56.92	72.55	3.52	5.46	14.55	25.41	3.91	28.90	21.66
SOLPA (HEAD)	64.43	77.76	81.08	86.36	**39.25**	**49.78**	53.94	70.08	**10.05**	**13.81**	21.77	35.75	3.86	30.82	22.75
Based on Qwen3-VL-8B-Instruct															
SOLPA (Direct)	54.78	69.51	74.96	82.40	21.45	29.05	53.37	69.60	3.41	5.30	16.78	28.73	4.06	23.36	21.61
SOLPA (HEAD)	**65.41**	**78.35**	**82.49**	**87.82**	21.94	28.51	**58.00**	**73.42**	7.10	10.66	**23.98**	**38.69**	**3.80**	**33.37**	**24.10**

test videos. Main two-stage planning results use predicted perception and annotated completed-step history. We use a 3B backbone on a fixed task-stratified subset for efficient component comparisons under matched training conditions. Oracle perception isolates Planner changes from recognition errors. Trajectory simulation separately evaluates planning with ground-truth state. Some source recordings span the released train and test partitions. After excluding 18 overlapping test videos, the existing 3B graph-on checkpoint scores 30.22% step-success (SS) and 21.91% PA on the remaining 498 videos (Section B).

Models and inference settings. We compare off-the-shelf multimodal large language models (MLLMs), published ProAct-Helper, and our Direct and two-stage variants. Direct jointly predicts trigger, task, step, future steps, and the assistant action from video. HEAD denotes the full two-stage system. Its Perceiver estimates the current state, and its Planner decodes actions from video, this estimate, and completed history. Unless otherwise stated, graph-on supplies graph text and a graph-derived action-name constraint, while graph-off omits both. Each pair fixes model parameters and, for HEAD, Perceiver predictions and annotated history. Off-the-shelf models receive no task-specific training or in-context video demonstrations. Table 1 distinguishes new and published scores.

Metrics and action selection. For perception, we retain the benchmark's accuracy (Acc), F1, mean per-class accuracy (mAcc), and macro-averaged F1 score (mF1) labels. Task/step Acc uses pooled Jaccard counts, while trigger Acc is ordinary accuracy (Section B). One-step planning uses step-success (SS) and parallel-action rate (PA), both in percent, and future-sequence edit distance (ED). SS checks the benchmark's successful remaining-step set. Our PA counts annotated parallel actions over all scored decisions, including Wait, whereas published ProAct-75 PA divides by effective non-Wait actions (Zhu et al., 2026). Cross-source PA values in Table 1 are therefore descriptive.

Table 2: **Optional task structure.** Graph-off removes graph text and its action-name constraint. Deltas: off − on.

HEAD inference		Planning metrics		
Scale	Mode	SS ↑	PA ↑	ED ↓
3B	on	30.37	22.60	3.85
	off	31.05 (+0.68)	22.14 (-0.46)	3.85 (0.00)
7B	on	30.82	22.75	3.86
	off	32.31 (+1.49)	22.38 (-0.37)	3.84 (-0.02)
8B	on	**33.37**	**24.10**	**3.80**
	off	32.30 (-1.07)	23.94 (-0.16)	3.82 (+0.02)

Table 3: **Training ablation.** The task-stratified 3B runs use oracle perception. H/G: history/graph. QA: current-state (Y1), completion-conditioned (Y2).

Methods				Graph-on (%)		Graph-off (%)	
H	G	QA(Y1)	QA(Y2)	SS ↑	PA ↑	SS ↑	PA ↑
–	–	–	–	–	–	26.45	16.42
✓	–	–	–	–	–	29.47	18.48
✓	✓	–	–	35.20	22.51	32.79	21.34
✓	✓	✓	–	30.97	23.16	32.18	21.54
✓	✓	–	✓	**36.50**	**26.35**	**38.26**	**26.28**

Our graph-mode pairs use identical rows and scoring. Our HEAD decodes its next action. Published ProAct-Helper filters predicted actions by the identified task's graph and applies an external entropy heuristic (Zhu et al., 2026). In trajectory simulation, SS is a raw scorer value in steps, and PA is the share of effective actions on a thread different from the human's most recent thread.

Implementation details. The 3B/7B models use Qwen2.5-VL (Bai et al., 2025b), and the 8B model uses Qwen3-VL (Bai et al., 2025a). Planners use rank-32 low-rank adaptation (LoRA) (Hu et al., 2021) for 10 epochs at a learning rate of 2×10^{-5}. We set $K = 12$, graph dropout $d = 0.7$, and $\lambda_{\text{QA}} = 0.10$. The QA-pool setting is 0.06; an auxiliary batch accompanies every planning minibatch. Inference is greedy, with no random graph omission. Full parameters and paired ablation intervals appear in Sections B and D.

4.2 PERCEPTION AND PLANNING PERFORMANCE

Table 1 compares perception and planning with graph-on for our models. The two-stage variants achieve higher task and step mAcc than published ProAct-Helper at both 3B and 7B. Task Acc, however, favors Direct at these sizes. Macro scores average classes, whereas task/step Acc pools confusion counts. Among our models, 8B leads in aggregate step F1 and 7B in macro step F1. Different pretrained backbones make these model comparisons rather than a parameter-only scaling study.

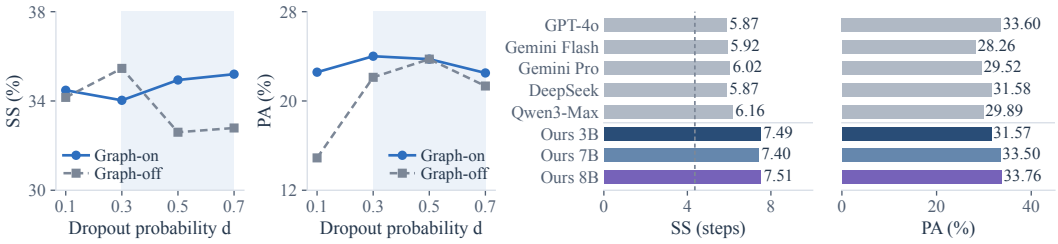
For model-decoded planning, HEAD lowers future-sequence ED and raises SS over same-backbone Direct in all three configurations, reaching 33.37% SS and 24.10% PA at 8B. This comparison evaluates the combined two-stage design and the annotated completed history supplied to HEAD. Component effects are examined in Section 4.3. Published ProAct-Helper obtains higher SS at 3B/7B (36.60%/36.10% compared with our 30.37%/30.82%). Those system-level scores include external graph-guided action selection, whereas Direct and HEAD are scored on model-decoded actions.

4.3 EFFECTS OF STRUCTURE-SUPERVISED TRAINING

Table 3 compares training components on the fixed 3B subset with oracle perception. Every row uses the ordinary planning targets defined in Section 3.2, including their graph-derived labels. The first row trains on video and annotated state without history, graph context, or auxiliary QA. Adding history (H) raises graph-off SS from 26.45 to 29.47 and PA from 16.42 to 18.48. Adding optional graph-context training (G) raises them further to 32.79 and 21.34. Thus, graph context during training benefits inference even when that context is later omitted.

H+G+Y1 tests whether adding current-state QA is sufficient. Compared with H+G, it raises graph-on/off PA by 0.65/0.20 points but lowers SS by 4.23/0.61 points. Current-state questions supervise recommendations for the observed state, but do not produce joint gains in successful step selection and parallel assistance here.

H+G+Y2 keeps the auxiliary pool size and training settings matched to Y1 while changing the question and target to reflect hypothetical completion. Relative to Y1, it improves graph-on/off SS by 5.53/6.08 points and PA by 3.19/4.74 points. In Figure 3, completing "Discard tissue" changes the



(a) Graph dropout: SS **(b) Graph dropout: PA** **(c) Trajectory SS** **(d) Trajectory PA**

Figure 4: **Training sensitivity and closed-loop assistance.** Shading in (a) and (b) marks $0.3 \leq d \leq 0.7$. Panels (c) and (d) use the ProAct-75 annotated-state simulator.

target to the already-eligible "Organize the mouse". These targets specify action reselection after hypothetical progress, including when no new action is unlocked. The matched comparison favors completion-conditioned questions and targets over current-state QA.

Relative to H+G, Y2 improves PA by 3.84 points in graph-on and 4.94 in graph-off. Paired video-bootstrap 95% intervals exclude zero for both gains, at $[0.99, 6.64]$ and $[1.06, 10.95]$, respectively (Section D). The graph-off SS gain is 5.47 points, with a wider interval of $[-0.27, 10.62]$. These intervals compare Y2 with H+G and quantify evaluation-video uncertainty for fixed checkpoints, excluding variation across training runs. The matched comparison between Y1 and Y2 reports point estimates.

The dropout sweep in Figure 4(a) and (b) uses H+G before completion supervision. Graph-off SS and PA peak at different dropout rates, so greater exposure to missing context does not improve both objectives monotonically. Settings and results appear in Section C.3.

4.4 Planning with optional procedural context

Table 2 compares graph modes with Planner parameters, Perceiver predictions, and annotated history fixed. Graph-off removes both graph text and its action-name constraint, so the Planner generates actions without a graph-derived vocabulary mask. It achieves 22.14–23.94% PA across the three models, with observed declines of 0.16–0.46 points. Graph-off SS increases at 3B/7B and decreases at 8B. These paired results support structure-optional action selection with the same parameters. In both modes, HEAD exceeds same-backbone Direct in SS and PA (Section C.1).

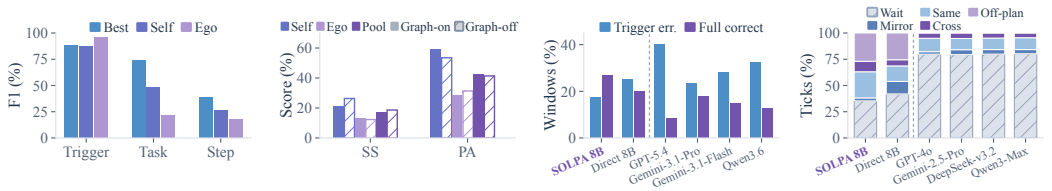
The training and inference comparisons test distinct roles of graph information. In the controlled 3B ablation with annotated state, adding optional graph-context training raises graph-off PA from 18.48% to 21.34% (Table 3). Both variants retain graph-derived targets. The main graph-mode comparisons instead fix Planner parameters, predicted state, and annotated history while removing graph text and its action-name constraint. These complementary results support using graphs during training while retaining action selection when explicit graph support is omitted at inference.

At 8B, Table 6 further varies the state input. With Perceiver output, graph-off HEAD achieves 32.30% SS and 23.94% PA. Annotated state raises these to 42.03% and 26.49%, while graph-on also improves. This inference-input ablation complements the 3B training ablation, showing graph-off planning under both state sources and the benefit of more accurate state information.

4.5 Closed-loop assistance

To test whether single-step choices accumulate into useful assistance, we evaluate the models in the benchmark's ground-truth-state simulator (seen in Figure 4(c) and (d)). Under the same graph-feasibility, alignment, and tie-breaking rules, the three SOLPA models obtain trajectory SS scores of 7.40–7.51, compared with at most 6.16 for the evaluated off-the-shelf large language models (LLMs). The no-robot score is 4.36 under this scorer, so the 8B result is 3.15 steps above that floor. Their trajectory PA rates of 31.57–33.76% lie within or slightly above the LLM range of 28.26–33.60%.

The 8B trajectory SS score exceeds the strongest LLM by 1.35 steps, while its 33.76% trajectory PA is close to GPT-4o's 33.60%. The 0.16-point PA gap is too small to support a strong ranking. Oracle selectors score 9.87 with privileged action selection, leaving room for improvement.



(a) OOD perception **(b) OOD planning** **(c) Perception errors** **(d) Decision outcomes**

Figure 5: **OOD and failure diagnostics.** Non-best-view perception and annotated-state planning appear in (a) and (b), while perception errors and simulator decision outcomes appear in (c) and (d).

The simulator supplies annotated state and permits actions that mirror the human. These results characterize assistance within its transition model rather than autonomous physical execution.

4.6 Analysis of Viewpoint Shift

To examine how viewpoint shift affects perception and structure-optional planning, we evaluate SOLPA 8B on non-best views from ProAct-75's self-collected videos and Ego-Exo4D. We assess recognition from video and compare graph-on/off planning with annotated states to separate planning performance from recognition errors.

As seen in Figure 5(a), task and step recognition remain challenging. Nevertheless, the Planner retains useful graph-off performance, as shown in Figure 5(b). Pooled PA is 41.38% in graph-off versus 42.33% in graph-on, with SS of 18.63% and 16.77%, respectively. Although graph-mode effects vary across sources, these results support structure-optional planning beyond the best-view setting under oracle perception. Improving cross-view state estimation could extend this capability to end-to-end assistance. On ten held-out tasks, a separate 3B Planner retains 24.44% PA without graph text, with annotated state and a shared task-specific action-name constraint (Section C.4).

4.7 Analysis of Failure Modes

We analyze perception errors and simulated decisions to distinguish recognition failures from ineffective assistance. Perception shares use each model's retained evaluation windows, while decision outcomes use all ticks in the 257-video annotated-state simulator (definitions in Section B). The LLM baselines use restricted remaining-step candidates, precluding off-plan actions.

Figure 5(c) shows a 17.51% trigger-error share for SOLPA 8B and 25.04% for Direct. Trigger, task, and step predictions are jointly correct in 26.74% of SOLPA windows and 19.78% of Direct windows. In Figure 5(d), SOLPA makes non-mirroring saved-step contributions on 34.9% of ticks and mirrors the person's next step on 2.8%. Direct records 20.7% and 11.9% on these outcomes, respectively. This breakdown highlights more useful assistance and less duplication, rather than greater action frequency alone. Better progress tracking could reduce legal off-plan choices that fail to advance the remaining work while preserving this collaborative preference.

5 Conclusion

We introduced SOLPA to learn proactive assistance from task graphs while keeping graph input optional at inference. Completion-conditioned targets supervise action reselection after hypothetical progress, and graph-context dropout supports both inference modes with shared parameters. The controlled 3B ablation favors completion-conditioned over current-state QA under matched data budgets and training settings, with gains in both planning metrics. Across three backbone configurations, the trained system supports graph-on and graph-off inference with fixed parameters and improves step recognition and planning over task-trained Direct on ProAct-75. Future work should improve cross-view perception and progress tracking over longer human–robot interactions.

AI USE STATEMENT

We used generative AI tools to assist with information retrieval, the discovery of relevant literature, and language editing of the manuscript. These tools helped improve clarity, grammar, and readability. The authors reviewed and revised the AI-assisted material as needed. The authors take full responsibility for the final manuscript and accompanying artifacts, including the accuracy of citations, interpretations, and reported findings.

ETHICS STATEMENT

This work uses the existing ProAct-75 benchmark to study video-based proactive assistance and evaluates action sequences in its annotated-state simulator. The auxiliary QA targets are derived from existing task graphs and annotations. Applying these models to physical assistance requires further assessment of action safety, robustness to unfamiliar situations, and the handling of visual data that may contain personal information. Appropriate human oversight and mechanisms for correcting inappropriate assistance remain important for deployment alongside people.

REPRODUCIBILITY STATEMENT

The method is described in Section 3. Supervision construction and loss definitions appear in Sections A.2 and A.3, with training and inference settings in Section B and input/output templates in Section E. The evaluation setup and trajectory simulation are specified in Sections 4.1 and 4.5, and Section D describes the uncertainty analysis. These details specify the inputs, targets, and evaluation conditions used for the reported experiments. Source code is available through the anonymous repository linked in the abstract.

REFERENCES

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

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

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pp. 287–318. Pmlr, 2023.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

Christopher A Field and Alan H Welsh. Bootstrapping clustered data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(3):369–390, 2007.

Matthew C Gombolay, Ronald J Wilcox, and Julie A Shah. Fast scheduling of robot teams performing tasks with temporospatial constraints. *IEEE Transactions on Robotics*, 34(1):220–239, 2018.

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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19383–19400, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE international conference on robotics and automation*, pp. 1470–1477. IEEE, 2011.

Pierre-Alexandre Kamienny, Kai Arulkumaran, Feryal Behbahani, Wendelin Boehmer, and Shimon Whiteson. Privileged information dropout in reinforcement learning. *arXiv preprint arXiv:2005.09220*, 2020.

Michail Karakikes and Dimitris Nathanael. The effect of cognitive workload on decision authority assignment in human–robot collaboration. *Cognition, Technology & Work*, 25(1):31–43, 2023.

Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence*, 38(1): 14–29, 2015.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, et al. Proactive agent: Shifting llm agents from reactive responses to active assistance. In *International Conference on Learning Representations*, volume 2025, pp. 47431–47457, 2025.

Kumaranage Ravindu Yasas Nagasinghe, Honglu Zhou, Malitha Gunawardhana, Martin Renqiang Min, Daniel Harari, and Muhammad Haris Khan. Why not use your textbook? knowledge-enhanced procedure planning of instructional videos. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18816–18826. IEEE, 2024.

Maithili Patel and Sonia Chernova. Proactive robot assistance via spatio-temporal object modeling. *arXiv preprint arXiv:2211.15501*, 2022.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pp. 4218–4227. PMLR, 2018.

Millicent Schlafly, Ahalya Prabhakar, Katarina Popovic, Geneva Schlafly, Christopher Kim, and Todd D Murphey. Collaborative robots can augment human cognition in regret-sensitive tasks. *PNAS nexus*, 3(2):pgae016, 2024.

Luigi Seminara, Davide Moltisanti, and Antonino Furnari. Viterbiplannet: Injecting procedural knowledge via differentiable viterbi for planning in instructional videos. *arXiv preprint arXiv:2603.04265*, 2026.

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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21064–21074. IEEE, 2022.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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 *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1207–1216. IEEE, 2019.

Marike Koch van Den Broek and Thomas B Moeslund. What is proactive human-robot interaction?-a review of a progressive field and its definitions. *ACM Transactions on Human-Robot Interaction*, 13(4):1–30, 2024.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

Haoyong Wu and Yongmei Liu. Prorac: A neuro-symbolic method for reasoning about actions with llm-based progression. *arXiv preprint arXiv:2511.15069*, 2025.

Honglu Zhou, Roberto Martín-Martín, Mubbasir Kapadia, Silvio Savarese, and Juan Carlos Niebles. Procedure-aware pretraining for instructional video understanding. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10727–10738. IEEE, 2023.

Fengming Zhu, Yuxin Pan, Xiaomeng Zhu, and Fangzhen Lin. A computable game-theoretic framework for multi-agent theory of mind. *arXiv preprint arXiv:2511.22536*, 2025.

Xiaomeng Zhu, Fengming Zhu, Weijie Zhou, Ye Tian, Zhenlin Hu, Yufei Huang, Yuchun Guo, Xinyu Wu, Zhengyou Zhang, Fangzhen Lin, et al. Proact: A benchmark and multimodal framework for structure-aware proactive response. *arXiv preprint arXiv:2602.03430*, 2026.

Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3532–3540. IEEE, 2019.

The appendix presents method details in Section A, implementation settings in Section B, and supplementary experiments in Section C. Statistical analysis and input/output templates follow in Sections D and E.

A SUPPLEMENTARY METHOD DETAILS

We detail local graph representation and decoding in Section A.1, planning targets and completion-conditioned QA in Section A.2, and training losses in Section A.3.

A.1 LOCAL GRAPH REPRESENTATION AND DECODING

The serializer records task and current-step names, local nodes with status tags, and edges. It also supports frontier, thread, and cross-thread candidate fields. The frontier lists visible next actions whose prerequisites would hold after the current step; other feasible actions can lie outside the local view. The fields reflect the renderer's available metadata. An unmapped thread uses `serial_main`, and an unavailable candidate list is empty. AND/OR conditions are used by the graph manager but are not serialized as separate activation expressions. QA suppresses the explicit cross-thread list while retaining the local graph and frontier fields. The local text is the Planner input, while the full task graph supplies offline labels and the graph-on action vocabulary. Representation diagnostics appear in Section C.5.

Constrained decoding. In the main graph-on setting, the decoder restricts action-name tokens to continuations compatible with the action vocabulary of the full task graph G. This vocabulary includes actions beyond the local feasible frontier. For prefix $u_{<j}$, let $\mathcal{C}_t(u_{<j})$ contain its allowed next tokens. The constrained distribution is defined as follows:

$$P^G_{w_d}(u_j \mid u_{<j}, x_t) = \frac{P_{w_d}(u_j \mid u_{<j}, x_t)\mathbb{1}[u_j \in \mathcal{C}_t(u_{<j})]}{\sum_{v \in \mathcal{C}_t(u_{<j})} P_{w_d}(v \mid u_{<j}, x_t)}. \tag{8}$$

This is a decoding constraint (De Cao et al., 2020), with no additional training. Vocabulary membership does not by itself certify prerequisite satisfaction or correct task identification. The main graph-off setting uses unconstrained generation with $C_t = \varnothing$.

A.2 PLANNING TARGETS AND COMPLETION-CONDITIONED QUESTIONS

Ordinary planning and completion-conditioned QA use graph-based labels with different candidate sets and conditioning states.

Ordinary planning targets. The ordinary planning target r^*_t is the cached `next_action` label produced by offline graph-based planning rules. These rules rank graph-feasible actions drawn from the annotated future sequence, falling back to feasible graph actions. This selection can include same-thread actions, while the QA teacher A_G considers only legal cross-thread candidates. The sequence π^*_t uses the graph manager's ordered remaining steps when available, otherwise the record's subsequent human-step annotations. This ordering specifies remaining task steps and can differ from the person's realized continuation. "Terminate" marks completion in the remaining-step format. These base targets are retained when graph context and auxiliary QA are omitted. Thus, H-only in Table 3 removes explicit graph input and QA, not graph-derived planning labels. The candidate completion u and recomputed QA answer $y^*_{t,u}$ belong to the separate auxiliary objective.

Completion questions and graph-based labels. The QA teacher A_G generates labels offline from task-graph rules, without training a separate model. The QA pool contains 2,034 completion-conditioned questions across 60 tasks. The main training recipe selects this type, called `counterfactual-unlock` in the data and Y2 in the paper. Current-state QA (Y1) serves as a training control in Table 3. It uses the same auxiliary pool size and training settings as Y2, with current-state questions and targets instead. Figure 3 illustrates both types. Y2 construction proceeds as follows:

- **Step 1:** Compute legal actions from G and D_t, excluding the human's current step. Group them using the graph manager's execution-thread assignments.

- **Step 2:** Select u as the first unfinished, non-current action from legal cross-thread actions, then other legal actions, and finally remaining steps. Legal candidates are ordered by action name.
- **Step 3:** Set $D'_t = D_t \cup \{u\}$, keep z_t^{step} fixed, and recompute legal cross-thread actions. Apply A_G to obtain $y^*_{t,u}$, or WAIT when no candidate remains.
- **Step 4:** Pair that answer with $q_t(u)$ and the local text view G_t^{QA} of the original state. The completion assumption appears in the question, not as an updated node status in the Planner's graph input.

The teacher uses the full task graph G, including AND/OR conditions and thread assignments. Its graph can contain actions absent from the Planner's local text G_t^{QA}. The remaining-step fallback makes the construction a symbolic completion query rather than an executed trajectory.

State-conditioned entropy score. We use the thread-mixing criterion of ProAct-Helper (Zhu et al., 2026) to rank the teacher's legal cross-thread candidates. For a query with completed set D and candidate a, let h_k count completed action nodes in thread k, and let b_k be one if a belongs to that thread and zero otherwise. The score is defined as follows:

$$H_{\text{mix}}(D, a) = \sum_k w_k \big[- p_k \log_2 p_k - (1 - p_k) \log_2 (1 - p_k) \big], \tag{9}$$

where $p_k = h_k/(h_k + b_k)$ and $w_k = (h_k + b_k)/\sum_j (h_j + b_j)$. Empty threads are omitted and $0 \log 0 = 0$. Completed actions initialize the human/world history. Robot history is empty before adding the candidate. For Y2, $D = D'_t$, including u in the completion counts. Structural nodes, "Terminate", and WAIT are not counted. The teacher minimizes this score, resolving ties within 10^{-9} by action-name order. Hence candidates on the same thread tie, and a previously untouched thread has zero mixing. The inherited score constructs labels offline. The Planner learns from these labels and decodes its own action at inference.

Worked completion example. In Figure 3, D_t contains collecting books and papers, placing batteries and glue on the shelf, and putting stationery into the pen holder. The human is placing book pages in a bookend. The current-state teacher selects "Discard tissue". Assuming its completion changes the target to "Organize the mouse", which was already eligible on another thread. The example illustrates reselection among remaining actions. Other queries make another action eligible or yield WAIT. The photographs are action references. The tissue image comes from a separate recording.

A.3 LOSS DEFINITIONS

For a serialized target y, let $\mathcal{T}(y)$ index its supervised tokens. The token loss is defined as follows:

$$\ell_w(y \mid x) = -\frac{1}{|\mathcal{T}(y)|} \sum_{j \in \mathcal{T}(y)} \log P_w(y_j \mid x, y_{<j}). \tag{10}$$

Prompt, padding, and masked prefix tokens do not contribute. Perception targets contain the annotated trigger, task, and step fields. Planning targets contain the future-step sequence and assistant action, with the annotated perception fields supplied as a teacher-forced prefix and masked from the planning loss. QA targets contain the teacher-selected action or WAIT.

We use the hierarchical binding loss of ProAct-Helper (Zhu et al., 2026). Let $\mathbf{h}_i^a$ denote the representation of output field $a \in \{\text{trig}, \text{task}, \text{step}\}$ in a trigger-positive example. A projection g maps the representations into a common d_{bind}-dimensional space. Write $\widetilde{\mathbf{h}}_i^a = g(\mathbf{h}_i^a)/\|g(\mathbf{h}_i^a)\|_2$ and $c_{ij}^{ab} = (\widetilde{\mathbf{h}}_i^a)^\top \widetilde{\mathbf{h}}_j^b$. For $N \geq 2$ valid pairs and temperature τ, we define the symmetric loss as follows:

$$\mathcal{B}(a, b) = -\frac{1}{2N} \sum_{i=1}^{N} \left[\log \frac{\exp(c_{ii}^{ab}/\tau)}{\sum_{j=1}^{N} \exp(c_{ij}^{ab}/\tau)} + \log \frac{\exp(c_{ii}^{ab}/\tau)}{\sum_{j=1}^{N} \exp(c_{ji}^{ab}/\tau)} \right]. \tag{11}$$

Thus $\mathcal{L}_{\text{trig2task}} = \mathcal{B}(\text{trig}, \text{task})$ and $\mathcal{L}_{\text{task2step}} = \mathcal{B}(\text{task}, \text{step})$. Matching fields from the same example form positive pairs, while fields from other examples provide negatives. These losses train the Perceiver independently of the Planner, with binding settings in Section B.

The Planner combines separately normalized planning and QA losses as defined in Equation (7). An auxiliary batch accompanies each planning minibatch, and the QA loader cycles as needed. The QA-pool setting controls the auxiliary pool size, not the frequency or weight of the QA loss. Both terms update the Planner's trainable parameters. Perceiver optimization is separate.

B TRAINING AND INFERENCE SETTINGS

Graph-context dropout applies to trigger-positive planning examples with a video-compatible task graph and renderable local text. QA examples retain their graph text. The Perceiver and Planner are trained separately. The 3B/7B models initialize from Qwen2.5-VL, and the 8B model from Qwen3-VL. Planner training starts from a fresh pretrained backbone rather than the trained Perceiver parameters. Adaptation uses LoRA together with stage-specific input embeddings and output modules; the remaining backbone parameters are frozen. Table 4 summarizes the main Planner recipe, which uses the 10-epoch checkpoint. Experiment-specific settings appear in Section C.

Table 4: **Main training and inference settings.** The QA-pool ratio determines pool size, while λ_{QA} weights the auxiliary loss.

Setting	Value
Planner optimization	
LoRA rank	32
LoRA α	32
LoRA dropout	0.05
LoRA targets	Attention query, key, value, and output; MLP gate, up, and down projections
Training duration	10 epochs
Learning rate	2×10^{-5}
Precision	BF16
Training devices	8
Per-device batch size	1 video chunk
Gradient accumulation	2 steps
Inputs and graph context	
Video context	Up to 5 extracted frames
Frame stride	3 extracted frames
History window K	12 completed steps
Future-step target	Remaining-step mode (annotated-sequence fallback)
Graph format	Compact YAML
Graph dropout d	0.7
Auxiliary supervision	
QA type	Completion-conditioned (Y2)
QA-pool ratio	0.06
QA loss weight λ_{QA}	0.10
QA update frequency	One auxiliary batch per planning minibatch
QA batch size	1
Inference	
Inference decoding	Greedy
Inference graph dropout	None
Perceiver generation limit	96 new tokens
Planner generation limit	192 new tokens
7B action-field recovery	Up to 32 new tokens for an incomplete action field
Graph-on decoding	Task-graph action vocabulary
Graph-off decoding	Unconstrained action-name generation

The Perceiver uses binding weights $\lambda_{\text{tt}} = 0.3$ and $\lambda_{\text{ts}} = 0.5$, projection dimension 256, temperature $\tau = 0.07$, and no additional trigger-discrimination loss. The Planner's perception prefix is teacher-forced during training, predicted by the Perceiver at inference, and annotated in oracle-perception evaluation. Completed history uses benchmark annotations. QA loss supervises answer tokens, masking prompt and graph tokens. Input/output templates appear in Section E.

Evaluation split check. Main results use the released benchmark split, in which some source recordings occur in both training and test partitions. Excluding the 18 affected test videos from evaluation of the existing 3B graph-on checkpoint gives 30.22% SS and 21.91% PA on 498 videos. This check evaluates the fixed checkpoint on a filtered test set.

Perception metric definitions. We retain the benchmark metric names in Table 1. Task and step Acc use $\text{TP}/(\text{TP} + \text{FP} + \text{FN})$ with pooled counts. This is a Jaccard score, not the fraction of correctly classified windows. Trigger Acc instead uses $(\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$. Here TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives. For each field, mAcc averages per-class Jaccard scores and mF1 averages per-class F1 scores. Aggregate F1 uses $2\text{TP}/(2\text{TP} + \text{FP} + \text{FN})$. The jointly correct shares in Figure 5(c) count windows directly and therefore differ from task/step Acc.

Simulator outcome definitions. Figure 5(d) partitions all decision ticks into five outcomes. Wait denotes no assistant action. Mirror duplicates the human's next action. Same and Cross denote non-mirroring saved-step contributions within and outside the human's execution thread, respectively. Off-plan denotes an in-task legal action outside the tick's remaining-step set, which does not advance the trajectory. The reported non-mirroring contribution share is Same plus Cross. All five shares use all decision ticks as the denominator, whereas trajectory PA is normalized over effective actions.

C Supplementary experiments

We first examine graph dependence in Direct (Section C.1) and inference inputs to the 8B system (Section C.2). We then analyze graph-context dropout (Section C.3), planning on held-out tasks (Section C.4), and graph-representation diagnostics (Section C.5).

C.1 Direct inference with optional task structure

Direct jointly predicts perception and planning outputs from video. Table 5 compares its graph-on/off modes at 3B, 7B, and 8B with parameters fixed within each pair. Graph-on supplies graph text and its action-name decoding constraint, whereas graph-off omits both. The HEAD results in Table 2 provide a same-backbone comparison under predicted perception.

Table 5: **Direct inference with and without graph guidance.** Parentheses give off − on.

Scale	Mode	SS (%) ↑	PA (%) ↑	ED ↓
3B	on	28.00	22.27	4.00
	off	25.80 (-2.20)	22.01 (-0.26)	4.13 (+0.13)
7B	on	28.90	21.66	3.91
	off	23.38 (-5.52)	20.09 (-1.57)	4.00 (+0.09)
8B	on	23.36	21.61	4.06
	off	16.88 (-6.48)	20.91 (-0.70)	4.41 (+0.35)

Switching Direct to graph-off increases ED and reduces SS and PA at all three scales. Its SS losses range from 2.20 to 6.48 points. For 3B Direct, a 0.26-point PA decline accompanies a 2.20-point SS decline, illustrating why both metrics matter when evaluating graph removal.

In graph-off mode, HEAD achieves higher SS and PA and lower ED than Direct at every scale. Its SS also exceeds graph-on Direct by 3.05, 3.41, and 8.94 points at 3B, 7B, and 8B, respectively. Its

PA advantage holds within each graph mode (Figure 1(d)). This comparison evaluates the combined training and input design, while Table 3 separately tests the auxiliary supervision.

C.2 GRAPH-MODE EFFECTS WITH PERCEIVER OUTPUT AND ANNOTATED STATE

Table 6 compares the 8B Direct system and one trained HEAD checkpoint under two state inputs and two graph modes. Direct generates trigger, task, step, future steps, and the next action jointly from video. For HEAD, Perceiver output supplies estimated trigger, task, and step, while annotated state substitutes their benchmark labels. Both HEAD rows use annotated completed history. Each graph-mode pair fixes the model parameters and state source while jointly changing graph text and the action-name decoding constraint. This is an inference-input ablation of the trained 8B system.

Table 6: **State and graph inputs at 8B.** HEAD receives Perceiver output or benchmark-annotated trigger, task, and step. Completed history is annotated in both HEAD rows. SS and PA are percentages. Parentheses give off − on.

	Graph-on			Graph-off		
Variant	SS ↑	PA ↑	ED ↓	SS ↑	PA ↑	ED ↓
Direct	23.36	21.61	4.06	16.88 (-6.48)	20.91 (-0.70)	4.41 (+0.35)
HEAD (Perceiver output)	33.37	24.10	3.80	32.30 (-1.07)	23.94 (-0.16)	3.82 (+0.02)
HEAD (annotated state)	42.53	27.56	2.54	42.03 (-0.50)	26.49 (-1.07)	2.54 (0.00)

With Planner parameters fixed, replacing Perceiver output with annotated state raises SS by 9.16 points in graph-on and 9.73 points in graph-off. The corresponding PA gains are 3.46 and 2.55 points. More accurate state input therefore benefits both configurations. Removing graph input reduces HEAD SS by 1.07 points with Perceiver output and 0.50 points with annotated state, while PA declines by 0.16 and 1.07 points. These input comparisons show that graph-off action selection remains useful under both state sources. They complement the controlled training comparisons in Table 3.

C.3 GRAPH-CONTEXT DROPOUT

We train 3B H+G models for 10 epochs on the task-stratified subset with $d \in \{0.1, 0.3, 0.5, 0.7\}$ and no auxiliary QA. Whole-context dropout retains video, annotated state, completed history, and planning targets. Each checkpoint is evaluated in both graph modes with oracle perception and no random omission at evaluation. Graph-off removes both graph text and its decoding constraint. Table 7 gives the values plotted in Figure 4(a) and (b). The $d = 0.7$ column matches H+G in Table 3.

Table 7: **Graph-context dropout sweep.** All 3B H+G runs use 10 epochs and oracle perception. Scores are percentages. Bold/underline mark the best/second-best scores for each metric and mode.

Metric	Mode	$d = 0.1$	$d = 0.3$	$d = 0.5$	$d = 0.7$
SS ↑	Graph-on	34.48	34.03	<u>34.94</u>	**35.20**
	Graph-off	<u>34.16</u>	**35.46**	32.60	32.79
PA ↑	Graph-on	22.58	**24.01**	<u>23.75</u>	22.51
	Graph-off	14.90	<u>22.12</u>	**23.75**	21.34

At $d = 0.1$, graph-off SS nearly matches graph-on (34.16% and 34.48%), but PA trails by 7.68 points. Preserving successful step selection therefore does not by itself preserve parallel assistance. The PA gap falls to 1.89, 0, and 1.17 points at $d = 0.3$, 0.5, and 0.7, respectively. Greater exposure to omitted context in these runs supports parallel action selection without graph input.

The preferred dropout rate depends on the metric. Graph-off SS peaks at $d = 0.3$, while graph-off PA peaks at $d = 0.5$, where both modes reach 23.75%. The fixed $d = 0.7$ setting gives the highest graph-on SS (35.20%) with a 1.17-point PA gap. This sweep characterizes the tradeoff in H+G training before adding the completion-conditioned supervision evaluated in Table 3.

C.4 PLANNING ON HELD-OUT TASKS

We test task-identity shift in a separate 3B Planner experiment. Ten task categories, five from COIN and five from Ego-Exo4D, are removed from both the video training split and the auxiliary Y2 QA data. This leaves 199 training videos from 64 observed task categories, compared with 240 videos before the holdout. The held-out evaluation manifest contains all 289 videos for those ten categories, with no video overlap with training.

We train only the Qwen2.5-VL-3B Planner with a frozen backbone and rank-32 LoRA for 10 epochs using eight GPUs, a global batch size of 8, a learning rate of 2×10^{-5}, and BF16. Training uses compact-YAML graph text with dropout $d = 0.7$, no completed-step history, and Y2 supervision with a QA-pool ratio of 0.06 and loss weight 0.10. The E-w2 evaluation supplies annotated trigger, task, and step. Each graph-on/off pair evaluates the same epoch-10 checkpoint.

Both modes constrain next-action decoding to all real action names in the full task graph plus "Wait / None". This vocabulary specifies the task's action set without filtering by current prerequisites or remaining steps. Graph-on/off here toggles graph text only, while the main comparison in Table 2 also switches the action-name constraint.

Table 8: **Task-identity holdout.** Both modes retain the same graph-derived action-name constraint and differ only in graph text. Rows use different data splits. Scores are percentages.

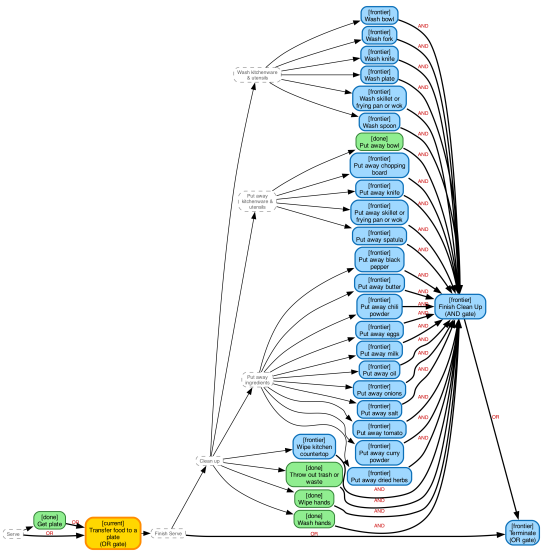
	Graph-on		Graph-off	
Evaluation set	SS ↑	PA ↑	SS ↑	PA ↑
Seen-task reference	30.84	21.67	26.61	24.92
Ten held-out tasks	29.26	25.15	26.94	24.44

On the ten held-out tasks, graph-on reaches 29.26% SS and 25.15% PA, while graph-off reaches 26.94% and 24.44% (Table 8). Omitting graph text lowers PA by 0.71 points and SS by 2.32 points. These results support assistance selection on unseen task identities without graph text when the task-specific action vocabulary remains available. The seen-task row provides context from a different training and evaluation split, rather than a matched estimate of the cost of task shift. Confidence intervals are unavailable for these pilot results.

C.5 GRAPH-REPRESENTATION DIAGNOSTICS

An annotated task graph can be presented as a rendered image or serialized text. Figure 6 places both views of a "Cooking Scrambled Eggs" state side by side. The rendering exposes cleanup branches and AND/OR joins spatially, while the text exposes exact node names, statuses, and edges. The text panel abbreviates the serialized format for readability. AND/OR gate labels in the image are not separate fields in the text view.

(a) Rendered task graph



(b) Text serialization (excerpt)

```
<|subgraph|>
task: Cooking Scrambled Eggs
human_now: Transfer food to a plate
nodes:
  - id: 623
    name: Get plate [done]
    status: done
    thread: serial_main
  - id: 634
    name: Transfer food to a plate [current]
    status: current
    thread: serial_main
  - id: 654
    name: Wash kitchenware & utensils [wrapper]
    status: wrapper
    thread: serial_main
  - id: 655
    name: Wash bowl [frontier]
    status: frontier
    thread: serial_main
  # Other nodes omitted.
edges: [[623, 634], [654, 655],
        [655, 1005]]
  # Other edges omitted.
frontier_names: ["Wash bowl"]
  # Other frontier names omitted.
legal_cross_thread_actions: []
<|/subgraph|>
```

Figure 6: **Rendered and serialized task graph for Cooking Scrambled Eggs.** The person is transferring food to a plate. The image displays AND/OR joins, while the text excerpt lists corresponding nodes, state tags, and edges.

Table 9: **Exploratory graph-representation diagnostics.** Accuracy, SS, and PA are percentages. Each group uses a separate evaluation protocol.

System	Input	Acc. (%)	SS (%)	PA (%)
	Node-label reading			
Qwen2.5-VL-3B	Rendered graph	0.0	–	–
Qwen2.5-VL-3B	Text serialization	22.6	–	–
OCR baseline	Rendered graph	5.3	–	–
	Graph questions (79 windows)			
GPT-5.4	Rendered graph	30.4	–	–
GPT-5.4	Text-only condition	21.5	–	–
	Oracle-perception planning			
Image-graph planner	Rendered graph	–	34.0	24.5
No-graph baseline	No graph	–	31.8	27.4

The node-label probe in Table 9 records 22.6% accuracy for Qwen2.5-VL-3B with text serialization and 0.0% with the rendered graph. This difference motivates exposing exact action names directly in text. A separate GPT-5.4 graph-question probe gains 8.9 points with the rendered graph, indicating that the utility of visual structure depends on the model and task. Image-graph planning improves SS by 2.2 points but lowers PA by 2.9 points against the no-graph baseline. These probes use separate protocols and support the choice of an accessible representation without establishing a universal advantage for graph images or text.

D Statistical analysis of the training ablation

We compare Y2 with H+G in Table 3 using 10,000 paired video-cluster bootstrap resamples (Field & Welsh, 2007). Videos are resampled as clusters, preserving their scored windows and using the same sampled videos for both models. Table 10 reports uncertainty in their score differences.

Table 10: **Paired uncertainty estimates for Y2 over H+G.** Gains and 95% confidence intervals are in percentage points. The bootstrap varies evaluation videos with trained models fixed.

Mode	Metric	Gain	95% CI	p-value
Graph-on	PA	$+3.84$	$[0.99, 6.64]$	0.0022
Graph-off	PA	$+4.94$	$[1.06, 10.95]$	0.0016
Graph-off	SS	$+5.47$	$[-0.27, 10.62]$	0.065

Both PA intervals exclude zero, supporting a benefit from adding Y2 supervision in both graph modes. The graph-off SS interval includes zero, so the statistical evidence is stronger for PA. These intervals quantify evaluation-video uncertainty for fixed checkpoints, not variation across training runs.

The matched Y1 and Y2 rows in Table 3 address a separate question about auxiliary content. Their point estimates compare current-state and completion-conditioned questions and targets under the same pool size and training settings. The bootstrap intervals above compare Y2 with H+G, not Y1. Together, these comparisons assess adding QA and choosing its content, while leaving changes to question wording and answer targets coupled.

E PROMPT AND OUTPUT TEMPLATES

The example in Figure 3 illustrates the shared system prompt (Section E.1), Perceiver inputs and outputs (Section E.2), and Planner prefix and continuation (Section E.3). Bracketed placeholders explain images and metadata, while node-status tags are literal. Ellipses omit nodes and edges.

E.1 SHARED SYSTEM PROMPT

Both stages use the following system message in the reported configuration. The completed-history and remaining-steps instructions are enabled by the corresponding training settings.

Shared System Prompt

```
You are a proactive video assistant.
Observe the scene, decide whether to trigger a proactive response,
and if so, predict the current task and step, upcoming human steps,
and the next robot action.
Output the following tags in order: trigger, task, step, future steps,
    next action.
Use none inside tags when information is unavailable.
Use the provided completed action history to avoid predicting already-
    done steps.
Predict all remaining human steps until task completion (include
    Terminate when the human is done).
Predict the next robot action.
End your response with <|im_end|>.
```

Here, "human steps" names the future-step field in the prompt; its supervised sequence follows the graph-manager ordering when available, as defined in Section 3.2.

E.2 PERCEIVER INPUT AND OUTPUT

The user message contains the video keyframes, video/frame metadata, a request to analyze the current scene, and latent episodic and structural-state slot tokens. It contains no task-graph text. Its content follows this order:

Perceiver w_p Input

```
Latent episodic memory:
<|ep_0|> <|ep_1|>
[video keyframe images]
Latent structural state:
<|str_0|> <|str_1|>
Video: TENCENT_T03S01A04cam01
Frames: [frame indices and timestamps]
Analyze the current scene and respond proactively.
```

The Perceiver receives no completed-history field and is supervised on the first three output fields. The example target is as follows:

Perceiver w_p Target Output

```
<|trigger_start|>true<|trigger_end|>
<|task_start|>Organize Desk<|task_end|>
<|step_start|>Place the book pages in a bookend<|step_end|>
```

E.3 PLANNER PREFIX AND CONTINUATION

The Planner receives the same keyframes and the recent completed-action history in its user message. For this state, the added history field is:

Planner w_d Completed-Action History

```
Human action memory:
- recent completed actions: Collect books and papers; Place the
    batteries and glue on the shelf; Put stationery into the pen
    holder
```

Its assistant prefix already contains the three state fields: Perceiver predictions at deployment, or annotated fields in oracle-perception evaluation. In graph-on mode, the serialized subgraph and two procedural placeholder tokens follow that prefix. The following excerpt shows their placement and the `compact_yaml` field format; omitted nodes and edges are not additional generated text.

Planner w_d Graph-On Assistant Prefix (Excerpt)

```
<|trigger_start|>true<|trigger_end|>
<|task_start|>Organize Desk<|task_end|>
<|step_start|>Place the book pages in a bookend<|step_end|>
<|subgraph|>
task: Organize Desk
human_now: Place the book pages in a bookend
nodes:
  - id: 4
    name: Place the book pages in a bookend [current]
    status: current
    thread: serial_main
  - id: 3
    name: Collect books and papers [done]
    status: done
    thread: serial_main
  ...
edges:
  - [3, 4]
  ...
<|/subgraph|>
<|proc_0|> <|proc_1|>
```

The procedural tokens are format placeholders, not a textual history. The Planner continues from this supplied prefix with the future task-step sequence followed by the assistant action. For this state, an example serialized continuation is:

Planner w_d Target Continuation

```
<|future_steps_start|>Terminate<|future_steps_end|>
<|next_action_start|>Discard tissue<|next_action_end|>
```

The displayed action is the current-state choice, matching Y1 in Figure 3. Y2 instead assumes tissue disposal and targets "Organize the mouse". The full serializer also writes frontier and cross-thread summary fields, omitted from this excerpt. The text graph is a local view, while graph-on action-name decoding draws its vocabulary from the full task graph. Graph-off omits the subgraph and its decoding constraint but retains the video, state prefix, and history.