EVENT2REWARD: LEARNING ROBOTIC REWARDS FROM PROGRESS RELATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reward models trained with normalized temporal labels from successful demonstrations can mistake setbacks for progress during exploration, turning losses of progress into positive shaping rewards that hinder policy improvement. Real suboptimal and failed executions offer corrective experience, but their absolute progress is difficult to label. We introduce Event2Reward, a reward model that learns from within-trajectory progress relations supported by task events in real executions. These relations directly supervise the progress predictions used for reward shaping. We combine them with temporal labels from successful demonstrations to anchor the numerical progress scale. We curate an event-annotated dataset from approximately 100K raw real-robot execution records across multiple sources and adapt Event2Reward using a small set of target-task executions spanning successful, suboptimal, and failed outcomes. On 40 tasks across four LIBERO suites, Event2Reward improves suite-average policy success over the strongest baseline in each suite by 9.96–13.76 percentage points under matched per-suite interaction budgets. In real-world experiments with a matched 30-minute wall-clock training budget, Event2Reward achieves an average success rate of 60.0% on three SERL insertion tasks, compared with 37.8% for Robometer, and 90.0% on three DSRL manipulation tasks, compared with 81.7% for Robometer. Ablations support the complementary roles of temporal labels and progress relations in downstream policy learning. We will release our model, dataset annotations, and processing tools.

1 INTRODUCTION

Reinforcement learning (RL) enables robots to improve manipulation skills through trial and error (Luo et al., 2025). However, learning on real robots often requires extensive and costly interaction. Reward shaping offers a way to accelerate policy learning by improving credit assignment with dense rewards. A common approach learns task progress from successful demonstrations, using normalized elapsed time as a proxy (Liang et al., 2026; Wang et al., 2026). This temporal supervision encodes monotonically increasing task progress. Progress estimates can serve as potentials for reward shaping, with the shaping term computed from discounted differences between successive predictions (Ng et al., 1999). During policy exploration, however, robots routinely produce suboptimal and failed executions, in which progress can stall or even regress. The reward model may still predict rising progress during such non-ideal behavior, assigning positive shaping rewards even when progress stalls or regresses. Instead of improving credit assignment, such rewards credit the actions that caused the loss, hindering rather than accelerating policy improvement.

A natural remedy is to train the reward model directly on suboptimal and failed executions. However, normalized time continues to assign increasing progress values during stalls and regressions, so it cannot provide valid absolute progress supervision for these executions. A common workaround repeats or reverses demonstration frames to synthesize stalls and regressions with labels known by construction (Liang et al., 2026; Wang et al., 2026), but these edits cannot introduce failure states absent from the original demonstrations. Another workaround uses outcome labels or trajectory-level preferences, which do enable reward learning from real suboptimal and failed executions (Lee et al., 2026; Liang et al., 2026). These signals describe overall execution quality, whereas progress-based shaping rewards depend on progress changes between adjacent observations. Such overall judgments therefore leave the timing and direction of these changes underdetermined: a failed execution

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 Event2Reward.** Left: pretraining data and real robot rollouts containing both successful and failed executions. Task events provide within-trajectory progress relations that complement temporal labels from successful demonstrations. Top center: task-conditioned progress predictions support reward shaping. Right: six real-world downstream tasks. The performance panels summarize real-world and LIBERO policy success, plug-insertion learning time, and the effect of rollout-based joint training. Plug-insertion time is measured until a rolling window of 10 autonomous training episodes reaches 100% success.

can retain partial progress, while a successful one can include regressions and recoveries. How, then, can we obtain local progress supervision from real executions without assigning an absolute progress value to every state?

Our key observation is that task events in real executions can establish local progress relations even when absolute progress values are unknown. For example, consider a robot moving a cup from a table onto a shelf. If the cup slips from the gripper and falls back onto the table, how much progress remains is difficult to quantify, yet this state clearly holds less task progress than the state just before the drop. We directly supervise the progress predictions used for reward shaping with these within-trajectory progress relations, which encode the direction of change between temporally anchored states. We combine these relations with normalized temporal labels from successful demonstrations to train our reward model, Event2Reward (Figure 1). The temporal labels anchor the numerical scale of predicted progress, while the relations constrain the direction of local progress changes in real suboptimal and failed executions.

We also contribute an event-annotated dataset curated from approximately 100K raw real-robot execution records across multiple sources, spanning successful, suboptimal, and failed executions. The annotations identify where task progress stalls, regresses, or recovers, and link these events to temporal regions and reference states. They turn observed execution events into local progress comparisons beyond terminal outcomes, without requiring framewise completion percentages. We combine two-pass VLM annotation with independent human annotation and review (Appendix A.3). We will release the structured annotations, data splits, and tools for constructing progress comparisons, providing a reusable resource for reward learning and execution analysis.

We use this supervision to train Event2Reward in stages. After pretraining on robot demonstrations and human videos, we continue training with temporal labels from normal successful executions and progress relations from event-containing rollouts. We adapt the reward model to downstream tasks using a few target-task demonstrations and event-containing executions under the same joint objective.

We evaluate whether Event2Reward improves policy learning on 40 LIBERO (Liu et al., 2023) tasks and six real-world tasks on Franka robots. The real-world experiments cover three precise insertion tasks trained with SERL (Luo et al., 2024; 2025) and three manipulation tasks trained with DSRL (Wagenmaker et al., 2025) using a π_0 (Black et al., 2024) base policy. Across the four LIBERO suites, Event2Reward improves average policy success by 9.96–13.76 percentage points over the strongest baseline in each suite under matched interaction budgets. In real-world experiments with a matched 30-minute training budget, Event2Reward increases average success from 37.8% to 60.0% on precise insertion tasks and from 81.7% to 90.0% on manipulation tasks, compared with Robometer, the strongest learned reward baseline in each setting. Our ablations show that combining temporal labels with within-trajectory progress relations yields larger gains in downstream policy learning than combining them with trajectory-level preferences. Joint training also outperforms either form of supervision alone in downstream policy learning, supporting the complementary roles of temporal labels and progress relations.

2 Related Work

Visual reward and value models. Learning rewards from visual observations reduces the need for task-specific reward engineering in robot learning. VIP (Ma et al., 2023b) learns goal-directed visual representations through value-based pretraining, and LIV (Ma et al., 2023a) extends this approach to language and image goals. For vision-language reasoning, GVL (Ma et al., 2024) estimates progress in context without parameter updates, while ProgressLM (Zhang et al., 2026) learns to assess progress relative to visual or textual demonstrations. WVM (Wang et al., 2026) instead uses video world-model representations to predict distributions over progress sequences. Our work focuses on the supervision of progress estimates: how real suboptimal and failed executions can provide evidence about local progress changes.

Progress supervision from demonstrations. Demonstrations provide temporal order and semantic task structure for progress supervision. Temporal order supports ranking-based objectives: Rank2Reward (Yang et al., 2024) learns rankings on expert videos and combines them with an expert–policy discriminator, while Video-Language Critic (Alakuijala et al., 2025) combines sequential ranking with video–text matching and uses failed executions as contrastive negatives. VLAC (Zhai et al., 2025) derives signed progress differences from temporal offsets. Task-stage annotations provide semantic structure: SARM (Chen et al., 2026) aligns numerical progress targets with annotated stages, and Robo-Dopamine (Tan et al., 2025) derives signed progress differences from stage-based progress coordinates. Augmentation extends this supervision to nonmonotonic sequences. ReWiND (Zhang et al., 2025) reverses demonstration segments to synthesize regressions, while WVM (Wang et al., 2026) constructs increasing, stable, and decreasing profiles through frame preservation, repetition, and reversal. Our approach complements these signals with progress relations grounded in stalls, regressions, and recoveries in real suboptimal and failed executions.

Reward learning from nonideal experience. Preference-based reward learning uses relative judgments without requiring absolute numerical ratings (Christiano et al., 2017). At the observation level, RL-VLM-F (Wang et al., 2024) asks a VLM to compare images, and Venkataraman et al. (2025) apply this approach to reward labeling for real-world offline RL. At the trajectory level, RoboReward (Lee et al., 2026) combines instruction relabeling and temporal clipping of successful demonstrations with naturally successful and failed rollouts carrying episode-level completion scores. Robometer (Liang et al., 2026) combines normalized temporal progress labels from expert demonstrations with success prediction and trajectory preferences, allowing real failures to contribute supervision while also producing per-frame progress estimates. Beyond these judgments, execution events provide structure for datasets and evaluation. WVM's Suboptimal-Value-Bench (Wang et al., 2026) converts annotated hesitation and retry intervals into reference progress curves, while REBOOT (Ofori-Ampofo et al., 2026) pairs precision-assembly demonstrations with phase boundaries, failure and recovery timestamps, and a phase-level evaluation protocol.

Our method uses execution events to directly supervise the progress predictions used for reward shaping. Event semantics, temporal anchors, and reference states specify supported within-trajectory relations without requiring a complete absolute progress curve for a failed execution. We

combine these relations with temporal labels from successful demonstrations to anchor the numerical progress scale.

3 METHOD

Event2Reward learns task progress from temporal labels and within-trajectory progress relations. After adaptation to downstream tasks, the model's scalar progress predictions serve as the potential for reward shaping. Figure 2 illustrates these complementary supervision signals.

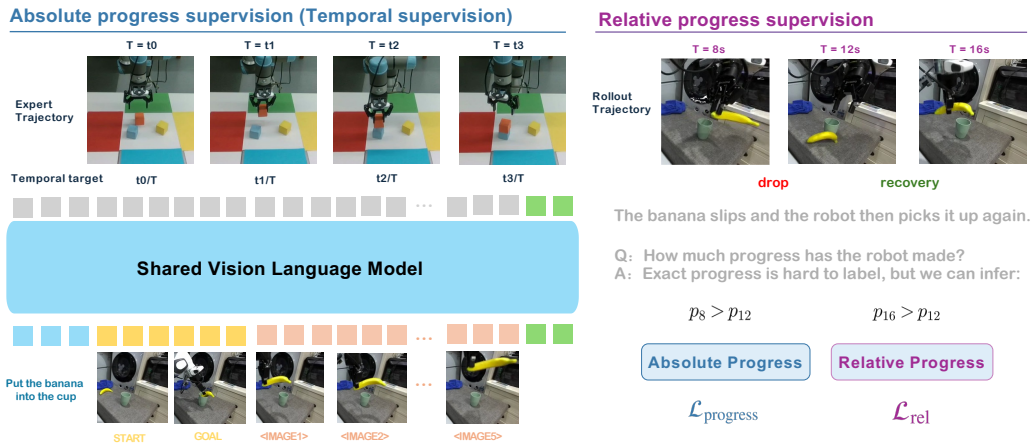


Figure 2: **Overview of progress supervision in Event2Reward.** Left: normalized temporal labels from successful expert trajectories anchor the progress scale of a shared vision-language model. Right: execution events in robot rollouts support relative constraints, illustrated by a drop and subsequent recovery ($p_8 > p_{12}$ and $p_{16} > p_{12}$). These complementary signals supervise the same progress model; relation supervision uses both successful and failed executions.

3.1 TASK PROGRESS MODEL

Event2Reward is a task-conditioned model that estimates task progress from visual observations. At time t, its input is $x_t = (\ell, o_0, H_t, g)$, where ℓ is the task instruction, o_0 is a reference start observation, H_t is a short visual history ending at the current observation, and g is an optional goal image depicting task completion. It outputs a scalar progress estimate $p_\theta(x_t) \in [0, 1]$, with higher values indicating greater progress toward completing the instructed task. The visual history provides context for interpreting local changes: progress may remain stable during stalled execution or decrease when previously achieved task states are lost.

We implement Event2Reward with a Qwen3.5-4B vision-language backbone, a progress head, and an auxiliary trend head. The progress head predicts a distribution over discretized progress values in $[0, 1]$, whose expectation gives $p_\theta(x_t)$. As detailed in Sections 3.2 and 3.3, temporal labels supervise this distribution, while within-trajectory progress relations constrain its scalar expectation. The same scalar serves as the potential for downstream reward shaping. The trend head predicts whether progress increases, remains stable, or decreases within the visual history as an auxiliary training task. Input construction and prediction-head details are given in Appendix B.1.

3.2 PROGRESS SUPERVISION FROM ROBOT EXECUTIONS

We curate an event-annotated dataset from approximately 100K raw real-robot execution records. The annotations connect observable execution events to temporal anchors and reference states, providing local progress comparisons beyond terminal success or failure. Normal successful trajectories supply normalized-time labels as a progress proxy over their task-relevant execution intervals. Trajectories containing events supply within-trajectory relations that capture stalled execution, loss of achieved task states, and recovery.

For each execution event, we record its category, a reference state, and the relevant temporal interval; regression-and-recovery events additionally identify a recovery state. Together with the supporting visual observations, these temporal anchors identify the within-trajectory progress comparisons used for supervision. Annotation therefore consists of identifying task events and their temporal anchors, without assigning framewise completion percentages.

Our annotation protocol organizes events by execution outcome, while their progress relations are grounded in observable task-state changes. For successful executions, we annotate *progress stall*, an interval with approximately unchanged task progress, and *regression and recovery*, a decrease in task progress followed by recovery to the pre-regression reference level. For failed executions, we distinguish *no further progress*, where no advancement beyond a reference state is observed before recording ends, and *unrecovered regression*, where task progress falls below its pre-regression reference level and does not return to that level before the recording ends. This organization does not imply that stalls or local recoveries occur only in successful executions. Event judgments depend on the task instruction and observable changes relative to the reference states. An unsuccessful grasp attempt is not regression unless a grasp had already been established, and renewed motion is not recovery unless the relevant task state is restored. A failed execution may retain partial progress, so failure alone does not establish unrecovered regression.

We construct event annotations through two-pass annotation with GPT-5.6 Sol and independent human annotation. GPT-5.6 Sol first identifies candidate events and temporal anchors across each execution, then refines them using denser local observations and later context when needed. Human inspection covers the raw real-robot collection, and each retained trajectory is independently annotated without access to the machine labels. Disagreements and unresolved cases undergo third-party review, while stratified samples of agreement cases are independently audited. Only visually supported progress comparisons are used for supervision. On the 8,146 retained RoboArena trajectories, machine and independent human annotations agree on the primary category for 87.56% of trajectories. Full annotation procedures and audit statistics are provided in Appendix A.3.

We convert the reviewed event annotations into within-trajectory progress comparisons. For each event e, its temporal interval locates candidate observations, and only those supported by the annotated task-state relation enter the comparison set $\mathcal{M}_e$. For example, when a dropped cup is regrasped and lifted, observations showing the loss of the established grasp should have lower progress than both the pre-drop reference and the recovered state. The annotations specify the observations to compare and the direction of their progress relation; the model learns the numerical scores through the relation loss in Section 3.3. Additional definitions and examples are provided in Appendix A.2.

3.3 Joint Training and Reward Shaping

Temporal supervision. For successful trajectories without annotated events, temporal labels are $y_t = (t-t_0)/(t_S-t_0)$ over the task-relevant interval from t_0 to completion t_S. We train the progress distribution with cross-entropy against a soft encoding with mean y_t, yielding $\mathcal{L}_{\text{progress}}$. During pretraining and replay, successful histories and their temporal transformations provide labels for the auxiliary trend cross-entropy loss $\mathcal{L}_{\text{trend}}$ (Appendix B.2). Reference and comparison observations used for relation supervision receive neither temporal nor trend labels.

Progress relations. Event categories select constraints on the scalar progress prediction $p_t = p_\theta(x_t)$, rather than serving as classification targets. Reference and comparison scores use the same model and task conditions; reference scores are recomputed as the model changes, with gradients stopped through them in the relation loss. For a regression-and-recovery event e, let $\bar{p}_a$ and $\bar{p}_r$ be the reference and recovery scores. For each $t \in \mathcal{M}_e$, we use

$$\ell_e(t) = \left[p_t - \min(\bar{p}_a, \bar{p}_r) + m_e\right]_+^2, \tag{1}$$

where $[z]_+ = \max(z, 0)$ and m_e is a margin determined by the training rule, not an annotated amount of progress loss. This penalizes insufficient separation below either reference. Progress stall permits a tolerance around the reference; no further progress penalizes scores above it; unrecovered regression requires a decrease margin without a recovery reference. We average the corresponding penalties within each event and then across events to obtain $\mathcal{L}_{\text{rel}}$. Full loss and margin definitions are given in Appendix B.3.

Joint objective. We jointly optimize $\mathcal{L} = \lambda_p \mathcal{L}_{\text{progress}} + \lambda_{\text{rel}} \mathcal{L}_{\text{rel}} + \lambda_t \mathcal{L}_{\text{trend}}$, with weights controlling the three contributions. Temporal labels anchor the numerical progress scale, while progress relations constrain local changes in real executions without assigning absolute targets to event observations. This distinction matters for reward shaping, whose magnitude depends on the prediction scale at a fixed shaping coefficient. Section 4.1 evaluates the complementary contributions of these two supervision signals.

Training and adaptation. We pretrain Event2Reward on approximately 20,000 hours of robot expert demonstrations and human manipulation videos using temporal labels and auxiliary trend supervision. We then continue training with the separately collected policy rollouts described in Section 3.2: normal successful rollouts provide temporal labels, while event-bearing rollouts provide within-trajectory progress relations. We also replay pretraining data during this stage to maintain numerical progress supervision. Finally, we adapt the reward model using target-task demonstrations and event-containing trajectories under the same joint objective. Data preparation and training settings are detailed in Appendix B.4, and adaptation protocols are given in Section 4.

Progress-based reward shaping. At reward inference, Event2Reward uses the inputs defined in Section 3.1; event categories and recovery annotations are not required. Its progress predictions provide the potential for shaping the environment reward (Ng et al., 1999):

$$r_t^{\text{shaped}} = r_t^{\text{env}} + \beta \left[\gamma p_{t+1} - p_t\right], \tag{2}$$

where r_t^{env} is the environment reward, γ is the RL discount factor, and β controls the shaping strength. The potential at absorbing terminal states is set to zero. Relation supervision trains this potential to reflect observed setbacks, targeting the misleading positive shaping signals described in the Introduction. The reward model remains fixed during policy learning in simulation and on real robots (Sections 4.2 and 4.4).

4 Experiments

We first use controlled ablations to compare supervision from within-trajectory progress relations and trajectory-level preferences, and to examine the complementary roles of progress relations and temporal labels. We then compare Event2Reward with baseline methods for policy learning in simulation. To assess the contribution of rollout-based joint training, we compare models under the same target-task adaptation budget. Finally, we evaluate whether Event2Reward improves policy success and learning efficiency on real robots.

4.1 Supervision Ablations

All reward-model variants use Qwen3-VL-4B-Instruct and start from the same checkpoint trained with temporal labels from normal successful LIBERO-10 rollouts. We vary the supervision during continued training and evaluate the resulting reward models through downstream policy learning on four LIBERO-10 tasks. Policy learning uses a residual SAC framework adapted from PLD (Xiao et al., 2026), with both the base policy and reward model frozen; predicted progress provides shaping rewards for residual policy learning. Within each task, all variants share the same downstream policy initialization and a budget of 1.2 million environment steps. Full protocols, statistical details, and additional comparisons are provided in Appendix C.1.

Supervision type. We compare temporal labels alone (*Progress-only*) with temporal labels combined with trajectory-level preferences (*Progress + Preference*) or within-trajectory progress relations (*Progress + Relation*). The latter two use the same additional trajectory pool and identical temporal supervision. We measure downstream policy learning using normalized evaluation AUC, which summarizes average evaluation success over a common interaction interval. Progress + Relation achieves the highest mean AUC on all four tasks, with a four-task average of 36.7%, compared with 19.6% for Progress-only and 19.4% for Progress + Preference (Figure 3). To distinguish the effect of relation supervision from that of mixed sampling, we also evaluate *Relation-loss-off*, which retains the mixed-sampling scheme of Progress + Relation but disables the relation loss. Its four-task mean AUC is 19.1%, close to Progress-only and below Progress + Relation (Table 10 in

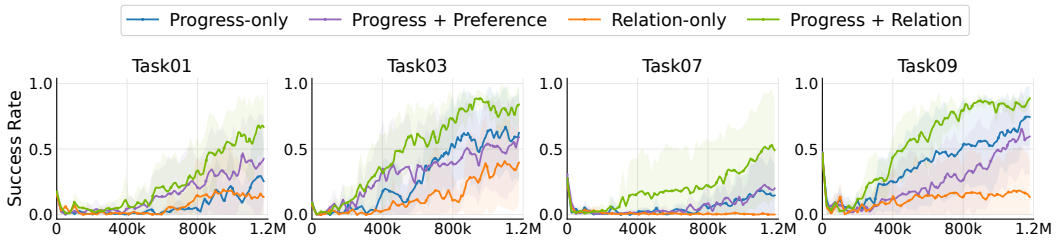


Figure 3: **Effect of reward supervision on policy learning.** Evaluation success on four LIBERO-10 tasks is plotted against environment steps. Lines show means across five environment seeds.

Appendix C.1). These results support a contribution from relation supervision beyond mixed sampling and show that adding progress relations to temporal labels yields larger downstream gains than adding trajectory-level preferences in this setting.

Joint supervision. To examine the complementary roles of temporal labels and progress relations, we compare Progress + Relation with Progress-only and *Relation-only*. The latter uses only progress relations during continued training. Joint supervision achieves higher mean evaluation AUC than either alternative on all four tasks (Figure 3). To investigate the weaker performance of Relation-only, we examine its progress predictions on held-out trajectories (Appendix C.1.2). On held-out regression trajectories, Relation-only achieves a higher relation satisfaction rate than both Progress-only and joint supervision. However, its progress predictions collapse to nearly identical values close to zero across trajectory stages. With the shaping coefficient fixed, these predictions would yield only weak progress-dependent shaping rewards. Joint supervision retains substantial variation in predicted progress while improving the relation satisfaction rate over Progress-only. Together, these results support the complementary roles of temporal labels in anchoring the numerical scale of predicted progress and progress relations in constraining local progress changes.

4.2 Policy Learning in Simulation

We compare Event2Reward with Sparse (environment rewards only), *Robo-Dopamine* (Tan et al., 2025) on four LIBERO (Liu et al., 2023) suites—Spatial, Object, Goal, and LIBERO-10—covering 40 tasks. Event2Reward's pretraining and joint training exclude LIBERO data. For each suite, Event2Reward and Robometer are adapted using the same 200 target-suite trajectories, while Robo-Dopamine follows its one-shot adaptation protocol with one expert demonstration per task. All reward models remain frozen during policy learning. Within each task, all conditions share the same frozen $\pi_{0.5}$ (Physical Intelligence et al., 2025) base policy and use the residual SAC framework from Section 4.1 under matched interaction budgets. Adaptation, policy-training, and evaluation details are provided in Appendix C.2.

Table 1: **RL performance on LIBERO.** Success rates (%) are mean ± sample standard deviation across five environment seeds. Each run contributes its last periodic independent evaluation within 200k environment steps for Spatial, Object, and Goal, and 1.2M for LIBERO-10.

Method	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-10
Sparse	36.80 ± 6.80	47.40 ± 3.13	66.36 ± 3.39	26.84 ± 8.80
Robo-Dopamine	36.72 ± 4.77	57.56 ± 6.09	64.00 ± 3.88	31.72 ± 4.92
Robometer	34.04 ± 5.03	60.56 ± 6.53	56.44 ± 5.67	40.88 ± 19.00
Event2Reward	$\mathbf{46.92} \pm 6.94$	$\mathbf{72.16} \pm 2.97$	$\mathbf{76.32} \pm 3.57$	$\mathbf{54.64} \pm 8.70$
Δ	**+10.12**	**+11.60**	**+9.96**	**+13.76**

Event2Reward achieves the highest mean policy success on all four LIBERO suites (Table 1). Compared with the strongest baseline in each suite, it improves success by 10.12, 11.60, 9.96, and 13.76 percentage points on Spatial, Object, Goal, and LIBERO-10, respectively.

4.3 Contribution of Rollout-Based Joint Training

We compare Event2Reward with a variant that skips rollout-based joint training (*w/o joint training*). Both start from the same pretrained checkpoint and use identical target-adaptation data, objectives, and update budgets. We evaluate them on LIBERO-Object's ten tasks using an average of 20 or 4 adaptation trajectories per task (the 200-trajectory set from Section 4.2 and its fixed 40-trajectory subset), with the same base policy, RL settings, and 200k-step interaction budget.

Table 2: **Effect of rollout joint training on LIBERO-Object.** Evaluation AUC and last periodic evaluation success (%) within 200k environment steps are mean ± sample standard deviation across five environment seeds, averaged over ten tasks.

	20 trajectories/task (avg.)		4 trajectories/task (avg.)	
Method	Eval AUC	Success	Eval AUC	Success
w/o joint training	29.67 ± 2.66	70.12 ± 9.51	27.22 ± 2.24	64.48 ± 7.16
Event2Reward	$\mathbf{33.06} \pm 2.28$	$\mathbf{72.16} \pm 2.97$	$\mathbf{31.34} \pm 2.58$	$\mathbf{67.04} \pm 5.47$
Δ	**+3.4**	**+2.0**	**+4.1**	**+2.6**

Rollout-based joint training improves mean evaluation AUC by approximately 3.4 and 4.1 percentage points with an average of 20 and 4 adaptation trajectories per task, respectively, and success by 2.04 and 2.56 points (Table 2). Mean task-level AUC improves on nine of ten tasks at each data size, supporting benefits even with fewer adaptation trajectories. This comparison evaluates the training stage as a whole, including its data mixture and additional optimization. Appendix C.2.4 provides detailed protocols and additional results.

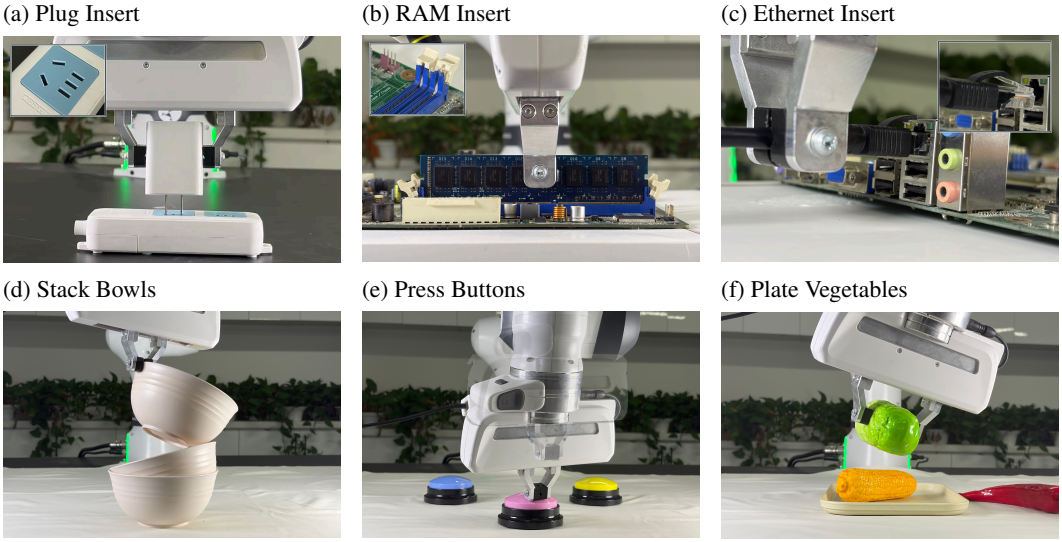


Figure 4: **Real-world tasks for policy learning.** Top: insertion tasks trained with SERL. Bottom: manipulation tasks trained with DSRL using a π_0 base policy. Insets in (a) and (b) show oblique views of the socket and RAM slots. The inset in (c) shows the Ethernet plug and port in an earlier frame before insertion. In Press Buttons, the translucent overlay depicts an earlier end-effector pose during the second press, while the opaque pose shows contact with the pink button.

4.4 Policy Learning on Real Robots

We evaluate Event2Reward on six real-world tasks using Franka robots: three precise insertion tasks trained with SERL and three manipulation tasks trained with DSRL using a π_0 base policy (Figure 4). Within each framework, we compare Event2Reward with Sparse, Robometer, and Robo-Dopamine. Event2Reward and Robometer use the same five successful and five suboptimal adaptation trajectories per task, while Robo-Dopamine follows its official one-shot adaptation protocol.

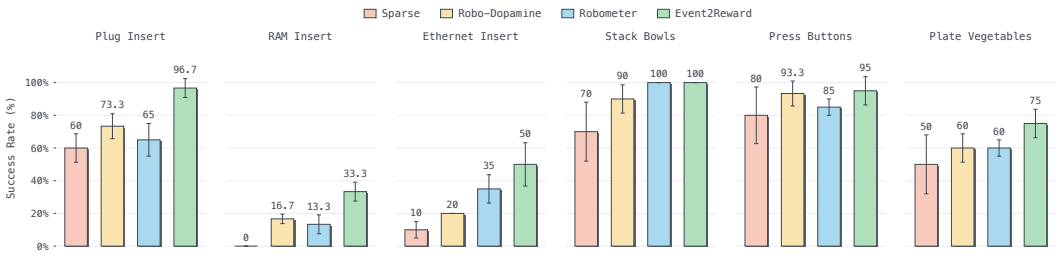


Figure 5: **Success rate (%) after 30 minutes of wall-clock training, including initial buffer warm-up where applicable, resets, and reward computation.** Bars show means and error bars show sample standard deviations across three independent training runs per task and reward condition. Each run's checkpoint at 30 minutes is evaluated over 20 autonomous episodes.

Within each task, reward models remain frozen during policy learning, and initialization protocols and learning settings are matched. We independently evaluate policies after 30 minutes of wall-clock training, including initial buffer warm-up where applicable, resets, and reward computation. Full protocols are provided in Appendix D.3.

Under this matched training budget, Event2Reward increases average success from 37.8% to 60.0% on precise insertion tasks and from 81.7% to 90.0% on manipulation tasks, compared with Robometer, the strongest learned reward baseline on average in each setting. These correspond to gains of 22.2 and 8.3 percentage points, respectively. Each task and reward condition is evaluated across three independent training runs, with each 30-minute checkpoint tested over 20 autonomous episodes. Setting-level averages weight the three tasks equally. Event2Reward achieves the highest mean success rate on five of the six tasks and ties with Robometer on Stack Bowls (Figure 5).

To complement the fixed 30-minute comparison, we conduct an additional plug-insertion training comparison with one run each for Event2Reward and Sparse under the same SERL framework. We measure the elapsed time until the most recent 10 autonomous training episodes are all successful for the first time. Event2Reward reaches this criterion in 15 minutes, compared with 45 minutes for Sparse, reducing the time to the criterion by two-thirds in this comparison. The window includes only autonomous episodes after warm-up; human-assisted initialization trajectories do not contribute to the success statistic. Both reported times include the initial buffer warm-up and environment resets.

5 Conclusion

We presented Event2Reward, a reward model that uses task events in real executions to learn progress-based shaping rewards. Temporal labels from successful demonstrations anchor the numerical scale of predicted progress, while within-trajectory progress relations constrain local changes during stalls, regressions, and recoveries. This formulation enables supervision from suboptimal and failed executions without requiring framewise absolute progress annotations. To support this learning framework, we curated an event-annotated dataset from approximately 100K raw real-robot execution records across multiple sources. Controlled ablations and progress diagnostics support the complementary roles of temporal and relational supervision: their combination improves downstream policy learning over either supervision alone and over temporal labels combined with trajectory-level preferences. With limited target-task adaptation, experiments on 40 LIBERO tasks and six real-world tasks demonstrate improved policy success under matched training budgets. The plug-insertion comparison further shows reduced training time to a common rollout-success criterion. Together, these findings establish execution events as a useful source of local progress supervision and demonstrate how real suboptimal and failed experience can contribute to effective reward learning for robotic manipulation.

AI USE STATEMENT

We used generative AI tools to assist with event annotation, data processing and analysis, code development for visualization, literature search and reference checking, and manuscript drafting, translation, and editing. Vision-language models proposed event annotations, with the annotation procedures and scope of human review described in Appendix A.3. LIBERO statistics were checked against recorded experimental evaluations. The authors take responsibility for the final content and conclusions of this work.

ETHICS STATEMENT

This work uses robot execution recordings and human manipulation videos to study reward learning for robotic manipulation. Event annotations for the real-robot rollout collection are produced through VLM-assisted annotation and human inspection of the full raw collection, with independent annotation and review procedures described in Appendix A.3. Annotation provenance, review status, and unresolved information are retained. During real-world SERL training, a human operator may terminate potentially unsafe motions. Our experiments evaluate task performance in the reported settings and do not establish safety for deployment beyond these settings.

REPRODUCIBILITY STATEMENT

We describe data sources, splits, event annotation, and human review in Appendix A, and model architecture, training objectives, and optimization settings in Appendix B. Appendix C provides ablation configurations and LIBERO reward-model adaptation, policy-learning, and evaluation protocols, including random seeds and metric aggregation. Real-world task definitions and training and evaluation procedures are described in Appendix D.

REFERENCES

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025. doi: 10.48550/arXiv.2503.06669. URL https://arxiv.org/abs/2503.06669.

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=jJOVpnNrEp.

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, Jonathan Tremblay, Kanav Arora, Kirsty Ellis, Luca Macesanu, Marcel Torne Villasevil, Matthew Leonard, Meedeum Cho, Ozgur Aslan, Shivin Dass, Jie Wang, William Reger, Xingfang Yuan, Xuning Yang, Abhishek Gupta, Dinesh Jayaraman, Glen Berseth, Kostas Daniilidis, Roberto Martin-Martin, Youngwoon Lee, Percy Liang, Chelsea Finn, and Sergey Levine. RoboArena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123v2*, 2025. doi: 10.48550/arXiv.2506.18123. URL https://arxiv.org/abs/2506.18123v2.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164v3*, 2024. doi: 10.48550/arXiv.2410.24164. URL https://arxiv.org/abs/2410.24164v3.

Boden Intelligence, Junpu Innovation Center, and MINT Lab, Shanghai Jiao Tong University. RW-RL dataset: Real-world reinforcement learning dataset. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/MINT-SJTU/RW-RL-Dataset.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations*, pp. 17091–17116, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1c92edb990a05f2269f0cc3afbb4c952-Abstract-Conference.html.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026. doi: 10.48550/arXiv.2605.02881. URL https://arxiv.org/abs/2605.02881.

Chengkai Hou, Kun Wu, Jiaming Liu, Zhengping Che, Di Wu, Fei Liao, Guangrun Li, Jingyang He, Qiuxuan Feng, Zhao Jin, Chenyang Gu, Zhuoyang Liu, Nuowei Han, Xiangju Mi, Yaoxu Lv, Yankai Fu, Gaole Dai, Langzhe Gu, Tao Li, Yuheng Zhang, Yixue Zhang, Xinhua Wang, Shichao Fan, Meng Li, Zhen Zhao, Ning Liu, Zhiyuan Xu, Pei Ren, Junjie Ji, Haonan Liu, Kuan Cheng, Shanghang Zhang, and Jian Tang. RoboMIND 2.0: A multimodal, bimanual mobile manipulation dataset for generalizable embodied intelligence. *arXiv preprint arXiv:2512.24653v3*, 2026. doi: 10.48550/arXiv.2512.24653. URL https://arxiv.org/abs/2512.24653v3.

Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576*, 2025. doi: 10.48550/arXiv.2509.00576. URL https://arxiv.org/abs/2509.00576.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945v2*, 2025. doi: 10.48550/arXiv.2403.12945. URL https://arxiv.org/abs/2403.12945v2.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint*

arXiv:2601.00675, 2026. doi: 10.48550/arXiv.2601.00675. URL https://arxiv.org/abs/2601.00675.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. In *Proceedings of Robotics: Science and Systems*, 2026. doi: 10.15607/RSS.2026.XXII.140. URL https://www.roboticsproceedings.org/rss22/p140.html.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16961–16969. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610040. URL https://doi.org/10.1109/ICRA57147.2024.10610040.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025. doi: 10.1126/scirobotics.ads5033. URL https://doi.org/10.1126/scirobotics.ads5033.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. *arXiv preprint arXiv:2306.00958v1*, 2023a. doi: 10.48550/arXiv.2306.00958. URL https://arxiv.org/abs/2306.00958v1.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=YJ7o2wetJ2.

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Jonathan Tompson, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. *arXiv preprint arXiv:2411.04549v1*, 2024. doi: 10.48550/arXiv.2411.04549. URL https://arxiv.org/abs/2411.04549v1.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999. URL https://people.eecs.berkeley.edu/~russell/papers/icml99-shaping.pdf.

Nana Yaw Owusu Ofori-Ampofo, Samira Ebrahimi Kahou, and Joseph Thekinen. REBOOT: From failure to recovery - a dataset and benchmark for precision assembly. *arXiv preprint arXiv:2609.22591*, 2026. doi: 10.48550/arXiv.2609.22591. URL https://arxiv.org/abs/2609.22591.

OpenLoong. Baihu dataset. Dataset release, 2025. URL https://ai.gitcode.com/openloong/baihu-dataset. Accessed: 2026-09-26.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz,

James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054v1*, 2025. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054v1.

Praveen Selvaraj, Lorenzo Uttini, and Ville Kuosmanen. ArmnetBench v0.1: Parallel real-world evaluation of manipulation policies on a low-cost arm farm. *arXiv preprint arXiv:2607.24481*, 2026. doi: 10.48550/arXiv.2607.24481. URL https://arxiv.org/abs/2607.24481.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-Dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703v1*, 2025. doi: 10.48550/arXiv.2512.23703. URL https://arxiv.org/abs/2512.23703v1.

Sreyas Venkataraman, Yufei Wang, Ziyu Wang, Navin Sriram Ravie, Zackory Erickson, and David Held. Real-world offline reinforcement learning from vision language model feedback. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13452–13459. IEEE, 2025. doi: 10.1109/IROS60139.2025.11246918. URL https://doi.org/10.1109/IROS60139.2025.11246918.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799v2*, 2025. doi: 10.48550/arXiv.2506.15799. URL https://arxiv.org/abs/2506.15799v2.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-F: Reinforcement learning from vision language foundation model feedback. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51484–51501. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24bn.html.

Zhihao Wang, Jianxiong Li, Yu Cui, Yuan Gao, Xianyuan Zhan, Junzhi Yu, and Xiao Ma. World value models for robotic manipulation. *arXiv preprint arXiv:2606.24742*, 2026. doi: 10.48550/arXiv.2606.24742. URL https://arxiv.org/abs/2606.24742.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, Shichao Fan, Xinhua Wang, Fei Liao, Zhen Zhao, Guangyu Li, Zhao Jin, Lecheng Wang, Jilei Mao, Ning Liu, Pei Ren, Qiang Zhang, Yaoxu Lyu, Mengzhen Liu, Jingyang He, Yulin Luo, Zeyu Gao, Chenxuan Li, Chenyang Gu, Yankai Fu, Di Wu, Xingyu Wang, Sixiang Chen, Zhenyu Wang, Pengju An, Siyuan Qian, Shanghang Zhang, and Jian Tang. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. In *Proceedings of Robotics: Science and Systems*. Robotics: Science and Systems Foundation, 2025. doi: 10.15607/RSS.2025.XXI.152. URL https://www.roboticsproceedings.org/rss21/p152.html.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, Zhaoye Long, Runtian Xu, Yue Wang, Chong Liu, Dihan Wang, Ziqiang Ni, Xiang Yang, You Liu, Ruoxuan Feng, Lei Zhang, Denghang Huang, Chenghao Jin, Anlan Yin, Xinlong Wang, Zhenguo Sun, Junkai Zhao, Mengfei Du, Mingyu Cao, Xiansheng Chen, Hongyang Cheng, Xiaojie Zhang, Yankai Fu, Ning Chen, Cheng Chi, Sixiang Chen, Huaihai Lyu, Xiaoshuai Hao, Yequan Wang, Bo Lei, Dong Liu, Xi Yang, Yance Jiao, Tengfei Pan, Yunyan Zhang, Songjing Wang, Ziqian Zhang, Xu Liu, Ji Zhang, Caowei Meng, Zhizheng Zhang, Jiyang Gao, Song Wang, Xiaokun Leng, Zhiqiang Xie, Zhenzhen Zhou, Peng Huang, Wu Yang, Yandong Guo, Yichao Zhu, Suibing Zheng, Hao Cheng, Xinmin Ding, Yang Yue, Huanqian Wang, Chi Chen, Jingrui Pang, YuXi Qian, Haoran Geng, Lianli Gao, Haiyuan Li, Bin Fang, Gao Huang, Yaodong Yang, Hao Dong, He Wang, Hang Zhao, Yadong Mu, Di Hu, Hao Zhao, Tiejun Huang, Shanghang Zhang, Yonghua Lin, Zhongyuan Wang, and Guocai Yao. RoboCOIN: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441v3*, 2026. doi: 10.48550/arXiv.2511.17441. URL https://arxiv.org/abs/2511.17441v3.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In *International Conference on Learning Representations (ICLR)*, pp. 13209–13236, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/163a875ab0c2631ed30b7435c4c9b8f1-Abstract-Conference.html.

Adina Yakefu, Bin Xie, Chongyang Xu, Enwen Zhang, Erjin Zhou, Fan Jia, Haitao Yang, Haoqiang Fan, Haowei Zhang, Hongyang Peng, Jing Tan, Junwen Huang, Kai Liu, Kaixin Liu, Kefan Gu, Qinglun Zhang, Ruitao Zhang, Saike Huang, Shen Cheng, Shuaicheng Liu, Tiancai Wang, Tiezhen Wang, Wei Sun, Wenbin Tang, Yajun Wei, Yang Chen, Youqiang Gui, Yucheng Zhao, Yunchao Ma, Yunfei Wei, Yunhuan Yang, Yutong Guo, Ze Chen, Zhengyuan Du, Ziheng Zhang, Ziming Liu, and Ziwei Yan. RoboChallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025. doi: 10.48550/arXiv.2510.17950. URL https://arxiv.org/abs/2510.17950.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2Reward: Learning shaped reward functions from passive video. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2806–2813. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610873. URL https://doi.org/10.1109/ICRA57147.2024.10610873.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026a. doi: 10.48550/arXiv.2602.15922. URL https://arxiv.org/abs/2602.15922.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. RoboFAC: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224v4*, 2026b. doi: 10.48550/arXiv.2505.12224. URL https://arxiv.org/abs/2505.12224v4.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, correct, and learn from manipulation failures via visual symbols. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 42386–42395, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zeng_Diagnose_Correct_and_Learn_from_Manipulation_Failures_via_Visual_Symbols_CVPR_2026_paper.html.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025. doi: 10.48550/arXiv.2509.15937. URL https://arxiv.org/abs/2509.15937.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 460–488. PMLR, 2025. URL https://proceedings.mlr.press/v305/zhang25a.html.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. ProgressLM: Towards progress reasoning in vision-language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11243–11271. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.516. URL https://aclanthology.org/2026.acl-long.516/.

Zhiyuan Zhou, Pranav Atreya, You Liang Tan, Karl Pertsch, and Sergey Levine. AutoEval: Autonomous evaluation of generalist robot manipulation policies in the real world. In *Proceedings*

of The 9th Conference on Robot Learning, volume 305 of *Proceedings of Machine Learning Research*, pp. 1997–2017. PMLR, 2025. URL https://proceedings.mlr.press/v305/zhou25a.html.

Table 3: **Source composition of the real-robot data.** Raw counts enumerate source records; Retained counts enumerate unique trajectories included in the training, validation, or test partitions. Both the outcome columns and the split columns sum to Retained. Uncertain denotes trajectories with usable local progress relations but unresolved terminal outcomes.

Source	Raw	Retained	Normal success	Suboptimal success	Failure	Uncertain	Train	Val.	Test
RoboChallenge	40,857	29,817	1,821	1,271	26,725	0	23,761	3,044	3,012
RW-RL	20,264	13,376	8,921	4,427	28	0	12,073	651	652
RoboArena	10,783	8,146	370	428	7,348	0	6,974	565	607
ViFailback	5,202	3,689	480	53	3,156	0	3,338	182	169
MolmoAct2	2,412	1,632	106	152	1,374	0	1,234	68	330
ArmNetBench	3,718	2,325	528	178	1,619	0	2,002	199	124
RoboFAC (real)	602	596	1	152	443	0	535	36	25
AutoEval	12,054	10,503	4,902	776	4,825	0	9,537	415	551
Other	3,306	2,508	800	226	1,440	42	2,237	136	135
Total	99,198	72,592	17,929	7,663	46,958	42	61,691	5,296	5,605

RoboFAC includes only its real-world subset. The raw ArmNetBench count includes 600 reference demonstrations, which are excluded from Retained. Of 73,084 trajectories with usable annotations, 492 are outside these partitions: 401 ViFailback benchmark trajectories and 91 trajectories from Other not included under the selected supervision and camera settings. Outcome categories follow the selected annotations; Failure includes executions that remain incomplete at the end of recording.

A Event-Annotated Robot Rollouts

A.1 Data Sources, Composition, and Splits

Data sources and scope. We curate an event-annotated dataset from approximately 100K raw real-robot execution records across multiple sources. These sources include collections for policy evaluation, real-world robot interaction, and manipulation-failure analysis. The public sources include RoboChallenge (Yakefu et al., 2025), RW-RL (Boden Intelligence et al., 2026), RoboArena (Atreya et al., 2025), ViFailback (Zeng et al., 2026), MolmoAct2 (Fang et al., 2026), ArmNetBench (Selvaraj et al., 2026), the real-world subset of RoboFAC (Ye et al., 2026b), and AutoEval (Zhou et al., 2025). We organize the visual observations, task instructions, and execution outcomes into a common format, and annotate task-progress events within the trajectories. The annotations link stalls, losses of achieved task states, and recoveries to temporal anchors, providing local progress comparisons for reward learning and execution analysis.

Execution quality and dataset composition. We distinguish normal successful, suboptimal successful, and failed executions. Normal successes complete the task without events identified by our annotation protocol, while suboptimal successes complete the task but contain such events. Failed executions remain incomplete at the end of the recording. A small number of retained trajectories have uncertain terminal outcomes but support local progress relations. Inclusion requires a task instruction, usable visual observations, and either a valid completion endpoint for normal-success progress supervision or at least one supported progress relation. The training, validation, and test partitions together contain 72,592 trajectories, with counts by source and execution quality reported in Table 3.

Data splits. We partition the dataset at the trajectory level, keeping each rollout and all its derived clips and event annotations in the same split. Records identified by a source as belonging to the same execution are grouped before splitting. The resulting splits contain 61,691 training, 5,296 validation, and 5,605 test trajectories. Validation and test trajectories are excluded from training, and training goal images are drawn only from the training split.

Table 4: **Execution events and the progress relations they support.** Comparisons apply to observations that support the annotated relation.

Execution event	Definition and supported progress relation
Progress stall	A local interval without observable task advancement; progress remains approximately unchanged relative to the reference state.
Regression and recovery	An achieved task state is lost and subsequently restored. Supported states during the loss have lower progress than both the reference and recovery states.
No further progress	The remaining execution provides no evidence of advancement beyond the reference state before recording ends. Supported states have progress no greater than the reference.
Unrecovered regression	An achieved task state is lost and remains unrecovered when recording ends. Supported post-loss states have lower progress than the reference.

A.2 Event Annotations and Examples

Annotation contents. Our annotations establish within-trajectory progress relations: comparisons of task progress between states within the same execution. For each annotated execution event, we record its category, a reference state, and the relevant temporal region; regression-and-recovery events also identify a recovery state. These temporal anchors and supporting observations determine which progress comparisons are justified. Where available, accompanying descriptions explain what task state had been achieved and how it changed. Execution events thus provide evidence for progress relations without requiring annotators to assign frame-level completion percentages.

Execution events and supported relations. Table 4 summarizes the four execution-event categories and the progress relations they support. Events are interpreted against the complete task instruction and observable task states. A failed outcome alone does not establish that progress decreased, since an execution may retain partial progress without completing the task.

These distinctions depend on task-state evidence rather than motion alone. Task-required waiting is not automatically a progress stall; an unsuccessful grasp attempt is not regression unless a grasp had been established; and renewed motion is not recovery unless the relevant task state is restored. No further progress does not require constant progress throughout the remaining execution, and unrecovered regression does not require a frame-by-frame monotonic decline.

An annotated interval locates an event; only observations supporting the stated relation are used for comparison. Later advancement or recovery is considered when interpreting the event, and comparisons with insufficient evidence are excluded. A locally supported relation can be retained even when the terminal outcome is uncertain. These relations supervise the progress model as described in Section 3.3 and Appendix B.3.

Illustrative annotations in simulation and on real robots. The following pages illustrate the four event categories in LIBERO (Liu et al., 2023) (pp. 19–22) and real-robot executions (pp. 23–26). Each domain contains four event groups per category, with seven frames per group, including the recording start and endpoint. The LIBERO and real-robot examples each cover 16 distinct trajectories. In this display, stall and recovery examples end in success, while no-further-progress and unrecovered-regression examples end in failure. These outcomes provide context; the supported progress relations depend on observable changes to task-relevant states rather than terminal outcome alone, and do not assign completion percentages.

Example sources and annotation provenance. The real-robot examples include seven successful executions from RoboChallenge (Yakefu et al., 2025) and nine executions from RoboArena (Atreya et al., 2025): one success and eight failures. All displayed real-robot examples use the final annotation records produced by the annotation and review procedure in Appendix A.3. The displayed LIBERO success events retain their existing human annotations. The LIBERO failure events re-

tain machine annotations and are not presented as human-verified. These examples illustrate event semantics and do not constitute an additional annotation-quality evaluation.

LIBERO: PROGRESS STALL

Progress stays approximately unchanged over $[a, r)$. Black borders mark the stall onset a and resumption r. All four videos finish successfully; each row shows F0, the stall, later progress, and the recording endpoint T.

(a) Put the yellow-and-white mug in the microwave and close it. The mug waits on the table, then is put inside and the door closes. **Stall: 21.6 s**



F0 / 0.0s Context | **F96 / 9.6s Start a** | F168 / 16.8s Stall | F270 / 27.0s Stall | **F312 / 31.2s Resume r** | F437 / 43.7s Later progress | F477 / 47.7s End T **Success**

(b) Put the black bowl in the bottom drawer and close it. After the pause, the bowl is placed inside and the drawer closes. **Stall: 8.8 s**



F0 / 0.0s Context | **F120 / 12.0s Start a** | F149 / 14.9s Stall | F190 / 19.0s Stall | **F208 / 20.8s Resume r** | F249 / 24.9s Later progress | F269 / 26.9s End T **Success**

(c) Turn on the stove and put the moka pot on it. The pot stays on the table during the pause, then reaches the lit stove. **Stall: 9.3 s**



F0 / 0.0s Context | **F170 / 17.0s Start a** | F201 / 20.1s Stall | F244 / 24.4s Stall | **F263 / 26.3s Resume r** | F306 / 30.6s Later progress | F326 / 32.6s End T **Success**

(d) Put both moka pots on the stove. The second pot stays on the table during the stall; both pots reach the stove afterward. **Stall: 15.0 s**



F0 / 0.0s Context | **F270 / 27.0s Start a** | F320 / 32.0s Stall | F390 / 39.0s Stall | **F420 / 42.0s Resume r** | F485 / 48.5s Later progress | F505 / 50.5s End T **Success**

Source: LIBERO; existing human annotations. Frames span F0 to the recording endpoint T (10 fps). W: wrist-camera view. Black borders mark annotated event anchors; green/red indicates success/failure.

LIBERO: REGRESSION AND RECOVERY

A previously attained task state is lost and restored. Black borders mark the reference a and recovery r; intermediate frames show the loss and regrasping. Each row begins at F0 and ends at the successful recording endpoint T.

(a) Put the alphabet-soup can and cream-cheese box in the basket. The can drops and is regrasped; both targets end in the basket.



F0 / 0.0s Context | **F71 / 7.1s Ref.** a | F81 / 8.1s Loss evidence | F176 / 17.6s Regrasp attempt | **F282 / 28.2s Recovery** r | F362 / 36.2s Later progress | F442 / 44.2s W End T **Success**

(b) Put the soup and tomato-sauce cans in the basket. The soup can is lost and regrasped; both cans reach the basket.



F0 / 0.0s Context | **F106 / 10.6s Ref.** a | F116 / 11.6s Loss evidence | F135 / 13.5s Regrasp attempt | **F164 / 16.4s Recovery** r | F249 / 24.9s Later progress | F334 / 33.4s W End T **Success**

(c) Put the cream-cheese box and butter in the basket. The butter slips during lifting, is grasped again, and joins the cheese box in the basket.



F0 / 0.0s Context | **F223 / 22.3s W Ref.** a | F233 / 23.3s W Loss evidence | F243 / 24.3s W Regrasp attempt | **F264 / 26.4s W Recovery** r | F300 / 30.0s Later progress | F320 / 32.0s W End T **Success**

(d) Put the soup and tomato-sauce cans in the basket. The tomato-sauce can drops, is lifted again, and moves into the basket.



F0 / 0.0s Context | **F213 / 21.3s Ref.** a | F223 / 22.3s Loss evidence | F240 / 24.0s Regrasp attempt | **F266 / 26.6s Recovery** r | F286 / 28.6s Later progress | F307 / 30.7s W End T **Success**

Source: LIBERO; existing human annotations. Frames span F0 to the recording endpoint T (10 fps). W: wrist-camera view. Black borders mark annotated event anchors; green/red indicates success/failure.

LIBERO: NO FURTHER PROGRESS

Black borders mark the annotated reference a after the last observed task advancement. Earlier frames provide context; later states do not advance beyond p_a. Each row begins at F0 and ends at the failed recording endpoint T.

(a) Put the bowl in the bottom drawer and close it. The bowl enters the drawer, but closing stops partway.



F0 / 0.0s	F207 / 20.7s	**F267 / 26.7s**	F330 / 33.0s	F393 / 39.3s	F456 / 45.6s	F519 / 51.9s
Context	Earlier progress	**Ref.** a	No advancement	No advancement	No advancement	End T
						Failure

(b) Put the bowl in the bottom drawer and close it. A short closing motion ends at the boundary; the drawer stays ajar.



F0 / 0.0s	F280 / 28.0s	**F340 / 34.0s**	F384 / 38.4s	F429 / 42.9s	F474 / 47.4s	F519 / 51.9s
Context	Earlier progress	**Ref.** a	No advancement	No advancement	No advancement	End T
						Failure

(c) Put the yellow-and-white mug in the microwave and close it. The mug is inside, but the door stops closing and remains ajar.



F0 / 0.0s	F218 / 21.8s	**F278 / 27.8s**	F338 / 33.8s	F398 / 39.8s	F458 / 45.8s	F519 / 51.9s
Context	Earlier progress	**Ref.** a	No advancement	No advancement	No advancement	End T
						Failure

(d) Put the mug on the plate and the pudding to its right. The pudding reaches the plate side but stays short of the required right-side placement.



F0 / 0.0s	F238 / 23.8s	**F298 / 29.8s**	F353 / 35.3s	F408 / 40.8s	F463 / 46.3s	F519 / 51.9s
Context	Earlier progress	**Ref.** a	No advancement	No advancement	No advancement	End T
						Failure

Source: LIBERO; machine annotations, not human verified. Frames span F0 to the recording endpoint T (10 fps). W: wrist-camera view. Black borders mark annotated event anchors; green/red indicates success/failure.

LIBERO: UNRECOVERED REGRESSION

An attained task state is lost and remains unrecovered. Black borders mark the reference a and loss ℓ. Each row starts at F0, shows the change from the attained state to the lost state, and ends at the failed recording endpoint T.

(a) Put both moka pots on the stove. A held pot falls behind the stove and remains on its side.



F0 / 0.0s Context | **F180 / 18.0s Ref.** a | F240 / 24.0s Before loss | **F250 / 25.0s Loss** ℓ | F260 / 26.0s After loss | F400 / 40.0s After loss | F519 / 51.9s End T **Failure**

(b) Turn on the stove and put the moka pot on it. The carried pot falls beside the stove and stays on its side.



F0 / 0.0s Context | F350 / 35.0s Before a | **F360 / 36.0s Ref.** a | **F362 / 36.2s Loss** ℓ | F368 / 36.8s After loss | F440 / 44.0s After loss | F519 / 51.9s End T **Failure**

(c) Put the white mug on the plate and the pudding to its right. The placed mug loses stable support and stays tilted.



F0 / 0.0s Context | F157 / 15.7s Before a | **F291 / 29.1s Ref.** a | **F298 / 29.8s Loss** ℓ | F300 / 30.0s After loss | F445 / 44.5s After loss | F519 / 51.9s End T **Failure**

(d) Put the yellow-and-white mug in the microwave and close it. The held mug falls and remains upside down outside the microwave.



F0 / 0.0s Context | F176 / 17.6s Before a | **F284 / 28.4s Ref.** a | **F294 / 29.4s Loss** ℓ | F304 / 30.4s After loss | F388 / 38.8s After loss | F519 / 51.9s End T **Failure**

Source: LIBERO; machine annotations, not human verified. Frames span F0 to the recording endpoint T (10 fps). W: wrist-camera view. Black borders mark annotated event anchors; green/red indicates success/failure.

REAL ROBOTS: PROGRESS STALL

Task progress remains approximately unchanged within the annotated interval ($p_t \approx p_a$). Later frames show renewed advancement.

(a) Move all task objects into the white box. The fork remains on the table during repeated grasp attempts. Grasping resumes, and all objects are in the box at the endpoint. **Stall: 26.4 s**



F0 / 0.0s Context | **F45 / 4.5s Start a** | F98 / 9.8s Stall | F164 / 16.4s Stall | **F309 / 30.9s Resume r** | F339 / 33.9s Later progress | **F1002 / 100.2s Success / T**

(b) Move all task objects into the white box. Repeated attempts leave the yellow brush on the table. Manipulation resumes, and the brush, fork and tape roll end inside the box. **Stall: 19.5 s**



F0 / 0.0s Context | **F39 / 3.9s Start a** | F78 / 7.8s Stall | F127 / 12.7s Stall | **F234 / 23.4s Resume r** | F264 / 26.4s Later progress | **F900 / 90.0s Success / T**

(c) Nest the three bowls into one stack. The blue bowl remains on the table during repeated approach attempts. The final frame shows all three bowls nested together. **Stall: 26.6 s**



F0 / 0.0s Context | **F160 / 16.0s Start a** | F213 / 21.3s Stall | F280 / 28.0s Stall | **F426 / 42.6s Resume r** | F869 / 86.9s Later progress | **F1023 / 102.3s Success / T**

(d) Nest the three bowls into one stack. Repeated attempts leave the green bowl outside the stack. The final frame shows the three bowls nested together. **Stall: 24.9 s**



F0 / 0.0s Context | **F286 / 28.6s Start a** | F336 / 33.6s Stall | F398 / 39.8s Stall | **F535 / 53.5s Resume r** | F565 / 56.5s Later progress | **F596 / 59.6s Success / T**

Source: RoboChallenge; existing human annotations. Black borders mark annotated event anchors. Green/red endpoints indicate success/failure. Frame numbers and times identify positions in each video. Some views are cropped to highlight task-relevant regions.

REAL ROBOTS: REGRESSION AND RECOVERY

An achieved task state is lost and then restored. Supported loss states have lower progress than both references ($p_t < p_a, p_t < p_r$).

(a) Place all loose fruits in the black wire basket. The green fruit is held, falls back onto the table, and is grasped again. All fruits are in the basket at the endpoint.



F0 / 0.0s Context | **F327 / 32.7s** **Ref.** a | F351 / 35.1s Loss evidence | F366 / 36.6s Reapproach | **F376 / 37.6s** **Recovery** r | F388 / 38.8s After recovery | **F1804 / 180.4s** **Success /** T

(b) Place all loose fruits in the black wire basket. The brown fruit slips from the grasp and is recovered. The final overview shows the fruits in the basket and the gripper withdrawn.



F0 / 0.0s Context | **F282 / 28.2s** **Ref.** a | F476 / 47.6s Loss evidence | F593 / 59.3s Reapproach | **F671 / 67.1s** **Recovery** r | F701 / 70.1s After recovery | **F3717 / 371.7s** **Success /** T

(c) Place all loose fruits in the rectangular wicker tray. The yellow fruit falls to the table and is regrasped. By the final frame, the fruits are collected in the tray.



F0 / 0.0s Context | **F185 / 18.5s** **Ref.** a | F197 / 19.7s Loss evidence | F234 / 23.4s Reapproach | **F246 / 24.6s** **Recovery** r | F261 / 26.1s After recovery | **F1527 / 152.7s** **Success /** T

(d) Put the banana into the cup. The lifted banana drops onto the counter, is grasped and lifted again, and is released into the cup.



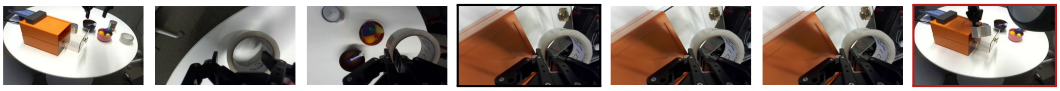
F0 / 0.0s Context | **F153 / 15.3s** **Ref.** a | F165 / 16.5s Loss evidence | F241 / 24.1s Reapproach | **F259 / 25.9s** **Recovery** r | F335 / 33.5s After recovery | **F400 / 40.0s** **Success /** T

Source: RoboChallenge (a–c), RoboArena (d). Examples use final annotation records from the procedure in Appendix A.3. Black borders mark annotated event anchors. Green/red endpoints indicate success/failure. Frame numbers and times identify positions in each video. Some views are cropped to highlight task-relevant regions.

REAL ROBOTS: NO FURTHER PROGRESS

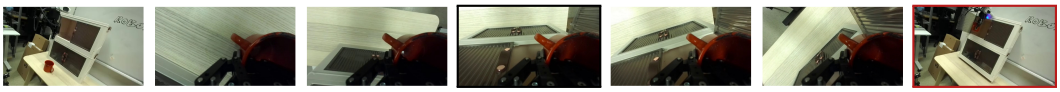
No later state advances beyond the annotated reference before recording ends ($p_t \leq p_a$). Images around a show the last advancement boundary.

(a) Place the tape roll inside the drawer. The roll is lifted to the drawer opening, then remains gripped outside without insertion or release.



F0 / 0.0s Start | F120 / 12.0s Before | F164 / 16.4s Before a | **F214 / 21.4s Ref.** a | F250 / 25.0s Plateau | F350 / 35.0s Plateau | **F400 / 40.0s Failure /** T

(b) Place the red cup inside the cabinet. The cup leaves the table and reaches the cabinet, then remains gripped outside through T.



F0 / 0.0s Start | F50 / 5.0s Before | F80 / 8.0s Before a | **F110 / 11.0s Ref.** a | F150 / 15.0s Plateau | F215 / 21.5s Plateau | **F265 / 26.5s Failure /** T

(c) Place the cup inside the white storage box. The cup is lifted, then stays held outside the white box without placement through T.



F0 / 0.0s Start | F250 / 25.0s Before | F300 / 30.0s Before a | **F330 / 33.0s Ref.** a | F350 / 35.0s Plateau | F375 / 37.5s Plateau | **F400 / 40.0s Failure /** T

(d) Place the plush toy inside the fabric bag. The toy is lifted toward the bag but remains gripped outside it while the arm repositions it.



F0 / 0.0s Start | F100 / 10.0s Before | F160 / 16.0s Before a | **F200 / 20.0s Ref.** a | F240 / 24.0s Plateau | F330 / 33.0s Plateau | **F400 / 40.0s Failure /** T

Source: RoboArena. Examples use final annotation records from the procedure in Appendix A.3. Black borders mark annotated event anchors. Green/red endpoints indicate success/failure. Frame numbers and times identify positions in each video. Some views are cropped to highlight task-relevant regions.

REAL ROBOTS: UNRECOVERED REGRESSION

An achieved task state is lost and remains unrecovered at the recording endpoint. Supported post-loss states have lower progress ($p_t < p_a$).

(a) Place both shoes inside the box. After one shoe is placed, the second slips from the gripper and stays outside the box at T.



F0 / 0.0s	F361 / 36.1s	**F373 / 37.3s**	F377 / 37.7s	F433 / 43.3s	F518 / 51.8s	**F799 / 79.9s**
Start	Before loss	**Ref.** a	Loss evidence	After loss	Later	**Failure /** T

(b) Place the banana on the plate. The banana is lifted, falls beside the empty plate, and remains on the table at T.



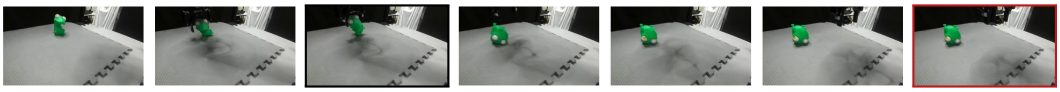
F0 / 0.0s	F57 / 5.7s	**F69 / 6.9s**	F72 / 7.2s	F81 / 8.1s	F129 / 12.9s	**F399 / 39.9s**
Start	Before loss	**Ref.** a	Loss evidence	After loss	Later	**Failure /** T

(c) Place the carrot inside the paper cup. The carrot falls from the raised gripper and is still on the table outside the cup at T.



F0 / 0.0s	F104 / 10.4s	**F116 / 11.6s**	F118 / 11.8s	F121 / 12.1s	F176 / 17.6s	**F400 / 40.0s**
Start	Before loss	**Ref.** a	Loss evidence	After loss	Later	**Failure /** T

(d) Lift the green frog from the table. The frog is lifted, slips out and rolls onto the mat, then remains ungrasped at T.



F0 / 0.0s	F156 / 15.6s	**F168 / 16.8s**	F169 / 16.9s	F180 / 18.0s	F228 / 22.8s	**F400 / 40.0s**
Start	Before loss	**Ref.** a	Loss evidence	After loss	Later	**Failure /** T

Source: RoboArena. Examples use final annotation records from the procedure in Appendix A.3. Black borders mark annotated event anchors. Green/red endpoints indicate success/failure. Frame numbers and times identify positions in each video. Some views are cropped to highlight task-relevant regions.

A.3 Annotation Procedure and Quality Control

Data scope and temporal alignment. We manually inspected the full collection of approximately 100K raw real-robot execution records. Records were organized by task instruction, source outcome, and observation timeline, with available camera views retained. Source success and failure outcomes were preserved. For successful rollouts used for progress supervision, we identified the task-relevant completion endpoint, retaining release and stabilization needed for completion while excluding unrelated post-task motion or waiting. RoboChallenge success videos additionally used a robot-state-derived execution timeline that compresses long static intervals while retaining context around motion boundaries. Annotations were aligned with the corresponding observation timeline, with mappings back to the source recordings.

Machine annotation. We used GPT-5.6 Sol with medium reasoning effort to generate initial annotations (V1) for all real-robot data sources from task instructions and indexed visual observations in two passes. A coarse pass identified candidate events, reference states, and temporal regions across the execution. A refinement pass used denser observations around these regions and later context when needed to refine event interpretation and boundary localization, including whether a lost task state was subsequently recovered. The annotations recorded event types and instances, temporal anchors, and brief descriptions of the supporting task-state changes.

Independent human annotation and review. Each retained trajectory had an independent human annotation (V2), produced without access to V1. We compared the two versions for event type, temporal order, instance count, and boundary localization, and checked the completeness of applicable information. Records with event or boundary conflicts, or unresolved required information, were referred for third-party review (V3). Reviewers first annotated independently without seeing either version, then compared their annotations with the frozen V1 and V2 records. Trajectories selected by stratified sampling from the set with no detected conflicts and no unresolved required information underwent independent auditing. Auditors were blinded to the original annotations and prior adjudication results.

Consolidation and final records. Final annotations incorporated confirmations and revisions from review and independent auditing, while the original machine and human records and review provenance were retained. Completion of review did not imply that every required field could be determined: unresolved information remained explicitly marked, and comparisons lacking sufficient support were excluded from supervision. Retained trajectories satisfied the eligibility criteria in Appendix A.1. The dataset composition statistics and real-robot examples in this paper use the final annotation records.

RoboArena audit example. We use RoboArena to illustrate the implementation and quality assessment of this shared procedure. The audit statistics in Table 5 concern its 8,146 retained trajectories, each independently annotated by one of ten human annotators. For this audit, corresponding event boundaries in V1 and V2 were mapped to a common processed observation timeline and expressed as normalized temporal positions within the effective trajectory interval. A boundary conflict was flagged for third-party review if any corresponding pair differed by at least 0.10 in normalized time. This criterion compares relative temporal positions rather than raw frame indices, accommodating annotations made at different frame rates. Independent audits of the concordant set were stratified by trajectory-level primary category. The regression-and-recovery and multiple-event success strata were checked in full, and trajectories were randomly sampled within the remaining strata. An audited trajectory was counted once as requiring revision if either frozen version needed correction.

A.4 Release Contents and Usage

Release contents. We will release the event annotations, data splits, source-media references, and tools for loading the data and constructing supervision. Each annotation remains linked to its source trajectory and relevant observations, enabling users to locate the reference state, event region, and recovery state when applicable. The records retain task instructions, execution outcomes, and annotation provenance, including human-review status. Where available, semantic descriptions explain

Table 5: **RoboArena annotation audit.** Statistics concern the 8,146 retained trajectories.

Audit item	Result
Primary-category agreement	7,133 / 8,146 (87.56%)
Third-party review	All 1,370 referred trajectories reviewed
Third-party conclusions	1,079 supported V2; 230 supported V1; 47 required revisions to both; 14 remained unresolved
Independent audit of the concordant set	800 / 6,776 trajectories inspected; 26 required revision (3.25% of the sample)
Stratified weighted revision rate	3.24%
Final annotation completeness	8,132 with complete applicable information; 14 with unresolved information

Primary-category agreement does not establish agreement on all event boundaries or required information. Revision rates concern the frozen annotations in the initial concordant set, rather than the final dataset's error rate. Unresolved records remain in the respective review and completeness denominators.

the task states and changes underlying an event. Together, these materials allow users to inspect the visual evidence and select observations for their own analyses. Detailed record formats and tool usage will be documented with the release.

Supervision construction and supported uses. In this work, normal successful trajectories provide temporal labels by normalizing time over their effective task intervals, while event annotations yield the within-trajectory progress relations described in Appendix A.2. The released tools will support reproducing this supervision construction; model inputs and training objectives are detailed in Appendix B. The accompanying descriptions support interpretation and retrieval and are not used as an additional training objective. The annotations can also support execution-error analysis and studies of whether progress estimates respond appropriately to progress stall, loss of achieved task states, and recovery. Researchers can use the underlying records to construct evidence-supported comparisons for alternative reward-supervision methods.

B Model and Training Details

B.1 Model Inputs and Prediction Heads

Inputs and visual history. Using the input components defined in Section 3.1, we process images at 224×224 resolution and take the start observation from the beginning of the trajectory or training segment. The instruction and images form a single multimodal input, with text markers indicating image roles and temporal order. The visual history contains five observations. For chronological inputs, it ends at the queried observation and contains no later observations. Videos in the event-annotated dataset use approximately 0.1-second sampling intervals, with each requested timestamp mapped to the latest available frame at or before it. Stage-1 pretraining and replay use 0.25-second intervals. AutoEval uses the latest five logged observations without assuming a fixed frame rate. When history is insufficient at the beginning of a trajectory or segment, we repeat the first eligible observation. Temporal transformations for trend supervision are described in Appendix B.2.

Goal conditioning. For the event-annotated dataset, goal images are selected from accepted completion frames of successful training trajectories with compatible task, scene, and camera context. Successful trajectories containing events may also supply goal images. When a compatible goal image is unavailable or omitted during training, we use a dedicated no-goal token. Goal-image dropout settings are given in Appendix B.4.

Backbone and prediction heads. With Qwen3.5-4B as the backbone, we append separate progress and trend query tokens to the multimodal input and apply a prediction head to each token's final-layer hidden state. Each head applies layer normalization followed by a two-layer MLP

with hidden width $d/2$, GELU activation, and dropout of 0.1 between the linear layers; d denotes the backbone hidden dimension. The progress head applies softmax to 101 logits, giving probabilities $\pi_{\theta,k}(x_t)$ over support values $k/100$ for $k = 0, \ldots, 100$. Its scalar prediction is the distribution's expectation:

$$p_\theta(x_t) = \sum_{k=0}^{100} \frac{k}{100}\, \pi_{\theta,k}(x_t). \tag{3}$$

The trend head applies softmax to three logits for increasing, stable, and decreasing progress.

B.2 Progress and Trend Supervision

Temporal labels. Successful demonstration segments used for pretraining and replay, and normal successful rollouts in the event-annotated dataset, provide temporal labels. Between the first eligible observation b and the accepted completion boundary S of the segment or rollout, we assign $y_t = (u_t - u_b)/(u_S - u_b)$, where u denotes time on the processed video timeline or the ordered observation index of a logged sequence. These labels serve as a numerical proxy for progress toward completing the accompanying instruction; observations after completion are excluded. Segment boundaries and timeline processing are described in Appendix B.4 for pretraining data and Appendix A.3 for the event-annotated rollouts. Replay retains the original pretraining segments and temporal labels. A zero-length interval receives only the completion label $y = 1$, without trend supervision.

Mean-preserving soft targets. To supervise the progress distribution in Appendix B.1 while preserving the scalar target, we encode $y \in [0, 1]$ using symmetric discrete Gaussian kernels over the 101 support values. With $K = 100$, define $\mathcal{S}_j = \{k \in \{0, \ldots, K\} : |k - j| \le \min(j, K - j)\}$ and

$$G_j(k) = \frac{\mathbf{1}[k \in \mathcal{S}_j] \exp\!\left(-\frac{(k-j)^2}{2\sigma^2}\right)}{\sum_{r \in \mathcal{S}_j} \exp\!\left(-\frac{(r-j)^2}{2\sigma^2}\right)}, \tag{4}$$

where $\sigma = 2$ bins. For $j_- = \lfloor Ky \rfloor$, $j_+ = \lceil Ky \rceil$, and $\alpha = Ky - j_-$, the soft target is

$$\widetilde{\pi}_k(y) = (1 - \alpha) G_{j_-}(k) + \alpha G_{j_+}(k). \tag{5}$$

The symmetric support and interpolation preserve expectation y; targets at zero and one remain concentrated at their respective endpoints.

Auxiliary trend targets. During pretraining, successful segments provide synthetic increasing, stable, and decreasing trend labels through the following sequences:

$$\begin{aligned} H_t^{\uparrow} &= (o_{t-4\Delta}, \ldots, o_{t-\Delta}, o_t), \\ H_t^{=} &= (o_t, o_t, o_t, o_t, o_t), \\ H_t^{\downarrow} &= (o_{t+4\Delta}, \ldots, o_{t+\Delta}, o_t), \end{aligned} \tag{6}$$

where Δ is the sampling interval. The decreasing sequence uses later observations from the source segment in reverse order as a training transformation. All three sequences end at o_t and retain the same progress target y_t; their trend labels follow the displayed order under the successful-segment progress proxy. Replay retains this construction. Pretraining endpoint queries and sequences lacking the temporal support required by a transformation are excluded from trend supervision.

For new normal successes in the event-annotated dataset, we use chronological histories and assign increasing labels only when all five observations come from distinct sampling positions, with no repetitions introduced by prefix padding. Histories containing repeated observations from prefix padding receive no trend label.

Losses and supervision scope. The progress loss $\mathcal{L}_{\text{progress}}$ is cross-entropy between the soft target $\widetilde{\pi}(y)$ and the predicted progress distribution. The trend loss $\mathcal{L}_{\text{trend}}$ is cross-entropy with the corresponding three-class label. Each loss is averaged over its own valid queries and is zero when none are available. Queries used for relation supervision, including reference and comparison observations, receive neither temporal labels nor trend labels. They supervise scalar progress through the relation losses in Appendix B.3.

B.3 PROGRESS-RELATION LOSSES

Reference predictions and supported comparisons. For each event e, let $\mathcal{M}_e$ be a nonempty set of sampled comparison observations supported by its annotation (Appendix A.2). The event provides a reference at a and, for regression and recovery, an additional recovery reference at r. For regression-and-recovery events, a marks the onset of loss while still representing the previously achieved task state, and r marks its restoration; the two anchors represent approximately equal task progress. Reference and comparison observations are evaluated under the same task instruction, start reference, and optional goal condition. Writing $p_t = p_\theta(x_t)$, we use $\bar{p}_a = \mathrm{sg}(p_a)$ and, when applicable, $\bar{p}_r = \mathrm{sg}(p_r)$, where sg stops gradients within the relation loss. This loss propagates gradients only through comparison predictions; reference scores are recomputed with the current model as its parameters change.

Relation penalties and margins. For $t \in \mathcal{M}_e$, we use the following squared hinge penalties, where $[z]_+ = \max(z, 0)$:

$$\begin{aligned} \ell_{\text{stall}}(t) &= \big[|p_t - \bar{p}_a| - \epsilon\big]_+^2, \\ \ell_{\text{regression}}(t) &= \big[p_t - \min(\bar{p}_a, \bar{p}_r) + m_e\big]_+^2, \\ \ell_{\text{no-progress}}(t) &= [p_t - \bar{p}_a]_+^2, \\ \ell_{\text{unrecovered}}(t) &= [p_t - \bar{p}_a + m]_+^2. \end{aligned} \tag{7}$$

Progress stall permits variation within ϵ of the reference score. Regression and recovery use the lower of the two reference scores, without requiring them to be equal. No further progress penalizes scores above the reference, while unrecovered regression also penalizes insufficient separation below it.

We set $\epsilon = 0.02$ and the base decrease margin $m = 0.1$. For regression and recovery, the effective margin is $m_e = \min\{m, \frac{1}{2}\min(\bar{p}_a, \bar{p}_r)\}$, reducing the required separation when reference scores approach zero in the bounded progress range $[0, 1]$. Unrecovered regression retains the fixed margin m. These tolerances and margins are optimization parameters, not annotated completion percentages or measured amounts of progress loss.

Loss aggregation. For the set $\mathcal{B}_e$ of valid event groups in a batch, we average comparison penalties within each event and then average over events:

$$\mathcal{L}_{\text{rel}} = \frac{1}{\max(1, |\mathcal{B}_e|)} \sum_{e \in \mathcal{B}_e} \frac{1}{|\mathcal{M}_e|} \sum_{t \in \mathcal{M}_e} \ell_e(t), \tag{8}$$

where ℓ_e is the penalty for the event type. The loss is zero when no valid event is present. Each sampled event has equal weight in this aggregation, regardless of its number of comparison observations. Source and event-type sampling proportions are given in Appendix B.4. We combine this loss with the progress and trend losses in Appendix B.2 using the joint objective in Section 3.3.

B.4 PRETRAINING AND JOINT TRAINING

This section describes pretraining and joint training of the full Event2Reward model. The controlled supervision ablations in Section 4.1 use a separate training configuration, detailed in Appendix C.1.

Pretraining corpus. Our pretraining corpus combines public robot demonstration datasets, additional robot expert demonstrations, and a private corpus of human manipulation videos. Table 6 reports the source composition, totaling approximately 20,000 hours. For DROID, we use the processed release provided by DreamZero (Ye et al., 2026a).

Training segments and temporal labels. We organize pretraining data into successful task or subtask segments, each paired with an instruction describing that segment. Segment boundaries follow available source annotations or delimit a complete successful execution. Source-specific preprocessing includes removing trailing inactivity and combining short adjacent human-video subtasks, together with their instructions, in temporal order. For DROID, we retain the DreamZero

Table 6: **Pretraining corpus composition.** Approximate durations sum the retained training segments after preprocessing and dataset splitting, before repeated sampling. These durations describe corpus composition rather than training sampling probabilities.

Source	Duration (h)
Robot demonstrations	
AgiBot World Beta (AgiBot-World-Contributors et al., 2025)	2,345
Galaxea Open-World (Jiang et al., 2025)	450
RoboMIND V2 (Hou et al., 2026)	319
RoboCOIN (Wu et al., 2026)	283
DROID (Khazatsky et al., 2025)	268
RoboMIND V1 (Wu et al., 2025)	190
Baihu: Qingloong subset (OpenLoong, 2025)	148
Other robot demonstrations	557
Robot subtotal	**4,560**
Human manipulation videos (private corpus)	15,487
Total	**20,047**

release's processed timeline, in which idle frames have already been removed. Temporal labels are normalized over each retained segment, so subtask segments measure progress toward completing the instructed subtask. Training queries use the inputs in Appendix B.1; the temporal labels and auxiliary trend supervision are defined in Appendix B.2.

Pretraining and initialization. We initialize the model from Qwen3.5-4B and pretrain it with the progress and auxiliary trend objectives described in Appendix B.2. Available goal images are omitted with probability 0.5. We initialize joint training from the Stage-1 checkpoint after 50,000 optimizer updates. The pretraining learning-rate schedule uses 25,000 warmup updates followed by cosine decay over a total horizon of 500,000 updates; the initialization checkpoint therefore precedes the end of this schedule.

Data sampling across training stages. During pretraining, robot demonstrations and human videos are sampled with probabilities of 70% and 30%, respectively. For joint training, we use the real-robot training split of the event-annotated dataset described in Appendix A, supplemented with 5,452 simulated rollouts from RoboFAC (Ye et al., 2026b).[1] Across the real and simulated data, the rollout training pools contain 17,572 normal successful trajectories and 49,571 event-bearing trajectories.

Replay uses a fixed subset of 147,456 successful task or subtask segments from the pretraining training set. These segments retain their original temporal labels, temporal transformations, and goal-image dropout. Robot and human replay data follow the same 70%/30% sampling ratio used during pretraining.

Joint training allocates 25%, 25%, and 50% of the model-query budget to replay, normal successful rollouts, and event-bearing rollouts, respectively. Both rollout pools include real and simulated executions. Normal successful rollouts provide temporal labels and valid trend labels, while event-bearing rollouts provide within-trajectory progress relations as defined in Appendix B.3. Available goal images in the rollout pools are retained without additional dropout.

Mixture proportions are measured in individual model queries, including event references and comparison observations. For each sampled event, we select at most eight supported comparison observations, together with its required reference observations. Event groups remain intact when constructing batches. Across the total query budget, progress stall, regression and recovery, no further progress, and unrecovered regression receive approximately 11.26%, 3.48%, 23.54%, and 11.73%, respectively. This sampling allocation determines event exposure; the relation loss itself uses the within-event and across-event averaging defined in Appendix B.3.

[1]The RoboFAC simulation subset also includes 292 validation trajectories, which are excluded from training. This subset is outside the real-robot collection reported in Appendix A.

Table 7: Pretraining and joint-training settings. Learning rates are base values before scheduling. Query counts include reference observations used in relation supervision.

Setting	Pretraining	Joint training
Checkpoint step	50,000	10,000
Scheduler horizon	500,000	40,000
Warmup updates	25,000	2,000
Global queries per update	256	80
Queries to checkpoint	12.8M	0.8M
Language learning rate	10^{-5}	2×10^{-6}
Vision learning rate	10^{-6}	5×10^{-7}
Prediction-head learning rate	2×10^{-5}	10^{-5}
Progress-loss weight λ_p	1	1
Trend-loss weight λ_t	1	1
Relation-loss weight λ_{rel}	0	10
Training seed	42	42

Optimization and training budget. Both stages update the language backbone, the final three layers of the vision encoder, the visual merger, and the two prediction heads. We use AdamW with weight decay 0.01, BF16 mixed precision, and gradient-norm clipping at 1.0. Separate learning rates are assigned to the language, vision, and prediction-head parameter groups. Both stages use linear warmup followed by cosine decay.

Table 7 summarizes the configuration. For joint training, we report the checkpoint after 10,000 optimizer updates, corresponding to 800,000 model queries. Its learning-rate schedule is defined over 40,000 updates. Checkpoint step and scheduler horizon are reported separately to specify the learning-rate trajectory that produced the model.

Validation and checkpoint reporting. We monitor joint training with a fixed query plan covering progress prediction on normal successes, consistency with annotated progress relations, and retention on Stage-1 data. The full plan contains 106,846 model queries, with a 12,096-query subset for intermediate monitoring. Goal-present and no-goal conditions are evaluated separately. The retention set consists of Stage-1 training segments held out from the Stage-2 replay pool. Test partitions are excluded from this monitoring plan.

The checkpoint reported here is identified by its 10,000-update training step. We report its validation measurements as diagnostics of progress calibration, consistency with annotated progress relations, and retention, without treating the end of the longer training schedule as evidence of a better model.

C Experimental Details and Additional Results

C.1 Ablation Study

Supervision conditions. We compare five reward-model variants on LIBERO-10 Tasks 1, 3, 7, and 9. All variants use Qwen3-VL-4B-Instruct and start from the same checkpoint trained for 2,000 updates using temporal labels from normal successful rollouts. Each variant then undergoes 2,000 additional updates with the supervision specified in Table 8. We preserve chronological input histories and disable auxiliary trend supervision throughout these ablations. Variant names refer only to the supervision used during continued training.

Training data. For the controlled supervision ablations, we use π_0 rollouts covering all ten LIBERO-10 tasks. We select 2,000 normal successful rollouts. This pool supplies temporal labels for the shared initialization and for continued training in variants that retain temporal supervision. An additional training pool contains 2,559 rollouts, comprising 839 event-containing successes and 1,720 failures. Relation and trajectory-preference supervision use this same additional pool.

The two training pools and the held-out validation and test sets, containing 128 and 139 rollouts respectively, are mutually disjoint at the trajectory level. Relation and preference supervision use

Table 8: Supervision used during continued training. All five variants start from the same checkpoint trained with temporal labels from normal successful rollouts. Relation-loss-off retains the mixed-sampling recipe of Progress + Relation but disables the relation loss.

Variant	Temporal labels	Trajectory preference	Progress relations
Progress-only	Yes	–	–
Progress + Preference	Yes	Yes	–
Relation-only	–	–	Yes
Progress + Relation	Yes	–	Yes
Relation-loss-off	Yes	–	–

the same trajectory splits. The held-out sets cover the same ten tasks as training, while downstream reinforcement learning evaluates Tasks 1, 3, 7, and 9.

Sampling and supervision. Temporal labels are constructed by normalizing time within normal successful segments. Progress-only uses clips from the normal-success pool throughout continued training, while Progress + Relation allocates half of its clip budget to this pool and half to event data. Event clips receive relation supervision and no temporal labels. Relation-loss-off retains the same 50/50 sampling recipe but sets the relation-loss weight to zero, controlling for the sampling change relative to Progress-only. Relation-only uses event data without temporal supervision; any padding clips carry no supervision. Under the fixed training budget, this condition therefore increases event exposure as well as removing temporal supervision.

Trajectory preference supervision. Progress + Preference uses the same normal-success pool and additional rollout pool as Progress + Relation. Preference supervision is derived from trajectory outcomes, without using event categories or boundaries. For preference supervision, we pair a successful and a failed trajectory from the same task and balance their presentation order. Each trajectory is represented by eight uniformly sampled frames, together with its start and goal images. The model jointly processes the two trajectories, and a separate preference head is trained with binary cross-entropy to favor the successful trajectory. Each update uses 40 normal-success clips and 14 trajectory pairs, matching the normal-success clip count and total image and visual-token exposure of Progress + Relation. Over 2,000 updates, this gives 80,000 normal-success clip exposures and 28,000 pair exposures. Relation supervision samples local event groups, whereas preference supervision samples whole-trajectory pairs. During downstream RL, the preference-trained model provides rewards through its progress head. Held-out evaluation of the comparison head confirms high accuracy on same-task success–failure preferences (Appendix C.1.1).

Optimization. We use AdamW with weight decay 0.01 and 100 warmup updates. Learning rates are 2×10^{-6} for the language backbone, 5×10^{-7} for the trainable vision parameters, and 10^{-5} for the prediction heads. We update the language backbone, the final three vision-encoder layers, the visual merger, and the heads used by each variant. The progress, relation, and preference losses have weights 1, 10, and 1, respectively, when active. The progress and relation objectives are defined in Appendices B.2 and B.3; auxiliary trend supervision is disabled here.

Additional baselines. We additionally evaluate the publicly released Robometer checkpoint (Liang et al., 2026) without further fine-tuning, and a Sparse baseline that uses only the environment reward. Our choice of Qwen3-VL-4B-Instruct matches the backbone used by Robometer. Robometer is an external reward-model baseline, separate from the Progress + Preference variant trained under our shared initialization and data protocol.

Residual reinforcement learning. We evaluate the reward models using a residual SAC framework adapted from PLD (Xiao et al., 2026). The $\pi_{0.5}$ base policy and reward model remain frozen throughout RL training. RL updates the residual actor, critics, and SAC temperature parameter, while the visual backbone remains frozen. Within each task, all conditions use the same base-policy checkpoint and residual architecture. At each decision step, the residual policy observes two camera

Table 9: Shared residual SAC settings for the supervision ablations. Replay entries correspond to action chunks.

Setting	Value
Optimizer	AdamW
Learning rate	3×10^{-4}
Network weight decay	10^{-4}
Gradient clipping norm	1.0
Replay capacity	250,000 chunks
Replay batch size	128 chunks
Target-network update coefficient	0.005
Initial SAC temperature	0.01
Actor hidden layers	3×256
Critic hidden layers	3×256
Number of critics	2
Actor and critic normalization	LayerNorm
Visual encoder	Pretrained ResNet-18
Visual backbone	Frozen

views, robot proprioception, and the proposed base-action chunk, and predicts corrections for the next $H = 5$ environment steps. We first compose the action chunk as

$$A_t = A_t^{\text{base}} + \alpha \Delta A_t, \qquad \Delta A_t \in [-1, 1]^{H \times 7}, \qquad \alpha = 0.1. \tag{9}$$

Corrections apply to all seven action dimensions. We then clip the gripper component of A_t to $[-1, 1]$ before executing the resulting action chunk. Network architecture and optimization settings are summarized in Table 9.

Progress-based reward computation. For learned-reward conditions, we evaluate the progress potential in Equation 2 at action-chunk boundaries. For a chunk containing $k \leq H$ executed environment steps, the training reward is

$$R_t^{\text{shaped}} = R_t^{\text{env}} + \beta \left[\gamma^k (1 - d_t) p_\theta(x_{t+k}) - p_\theta(x_t) \right], \tag{10}$$

where R_t^{env} is the accumulated environment reward within the chunk, $\gamma = 0.998$, and $\beta = 1$. The indicator d_t is one at an episode termination or truncation boundary and zero otherwise; the next-state potential contribution is therefore zero at these boundaries. SAC bootstrapping uses the same duration-dependent discount and episode-boundary mask. Reward inference runs asynchronously with environment interaction; transitions enter the replay buffer after their shaped rewards have been computed. Sparse uses environment rewards alone.

Optimization and interaction budget. RL updates use only online experience, with no offline replay or critic pretraining. Updates begin after the replay buffer contains 1,000 chunk transitions. Each learner iteration performs one critic update on a 128-transition batch, followed by four critic updates on 32-transition minibatches and one actor and temperature update on the corresponding full batch. Each run receives a budget of 1.2 million environment steps. We use five environment seeds, $\{7, 8, 9, 17, 27\}$, for every task and condition.

Policy evaluation. We evaluate policy snapshots in a separate environment every 50 completed training episodes, using 50 evaluation episodes per snapshot. Evaluation uses deterministic residual actions while retaining the base policy's default stochastic sampling. The initial evaluation sets all residual actions to zero and measures base-policy performance. Evaluation success is the fraction of successful episodes according to the environment's task-success criterion; evaluation trajectories are not added to the training replay. The main figure reports these independent evaluations against environment interaction steps. Supplementary rollout curves report the success rate over the most recent 100 completed training episodes, starting once 100 episodes are available. Episodes interrupted by the interaction-budget cutoff are excluded from this rollout statistic.

Table 10: **Complete RL results on four LIBERO-10 tasks.** Values are percentages, reported as mean ± sample standard deviation across five environment seeds. *Average* first averages the four tasks within each seed. Bold indicates the highest mean in each column within each panel.

Method	Task 1	Task 3	Task 7	Task 9	Average
(a) Normalized evaluation AUC					
Sparse	1.4 ± 1.0	30.4 ± 29.6	3.5 ± 5.3	24.5 ± 30.9	14.9 ± 8.1
Robometer	22.1 ± 19.8	26.5 ± 15.2	5.6 ± 9.1	29.1 ± 22.8	20.8 ± 11.9
Progress-only	6.6 ± 7.8	29.4 ± 18.5	5.5 ± 8.6	37.0 ± 25.4	19.6 ± 10.1
Progress + Preference	16.1 ± 18.7	31.0 ± 29.1	6.0 ± 9.0	24.4 ± 23.2	19.4 ± 14.3
Relation-only	6.8 ± 12.8	13.4 ± 15.4	1.1 ± 0.2	11.8 ± 23.4	8.3 ± 7.9
Relation-loss-off	13.2 ± 16.3	30.4 ± 22.1	5.7 ± 7.0	27.2 ± 17.7	19.1 ± 7.6
Progress + Relation	$\mathbf{24.2} \pm 15.7$	$\mathbf{48.9} \pm 14.3$	$\mathbf{19.8} \pm 26.3$	$\mathbf{53.8} \pm 8.8$	$\mathbf{36.7} \pm 7.8$
(b) Last periodic evaluation success					
Sparse	4.0 ± 7.9	49.2 ± 44.3	15.6 ± 32.7	39.6 ± 48.2	27.1 ± 24.4
Robometer	46.0 ± 42.2	70.4 ± 27.2	19.6 ± 36.2	62.4 ± 34.7	49.6 ± 21.7
Progress-only	27.2 ± 36.9	69.6 ± 38.6	14.8 ± 28.7	72.8 ± 25.3	46.1 ± 15.2
Progress + Preference	49.6 ± 39.8	57.2 ± 35.1	22.8 ± 39.0	58.0 ± 32.9	46.9 ± 30.5
Relation-only	14.8 ± 33.1	39.2 ± 36.6	0.0 ± 0.0	17.6 ± 39.4	17.9 ± 14.6
Relation-loss-off	39.6 ± 38.4	67.2 ± 29.9	27.2 ± 34.7	78.0 ± 8.6	53.0 ± 14.8
Progress + Relation	$\mathbf{66.4} \pm 28.5$	$\mathbf{88.8} \pm 12.9$	$\mathbf{52.8} \pm 48.6$	$\mathbf{90.8} \pm 5.0$	$\mathbf{74.7} \pm 14.2$

Evaluation and curve aggregation. Following the evaluation protocol above, we align curves by linear interpolation between recorded observations, without extrapolation or additional smoothing. Curves show equally weighted means across five environment seeds, with shaded bands indicating the mean $\pm 1.96\,\mathrm{SEM}$. Here, SEM is the sample standard deviation across seeds divided by $\sqrt{5}$, and bands are clipped to the valid success-rate range. Each learned-reward condition uses the same reward-model checkpoint across environment seeds; the bands therefore summarize variation across downstream runs. Result tables report means ± sample standard deviations across seeds.

Summary metrics. We report normalized evaluation AUC and success at the last periodic evaluation. For a given task and condition, let $\widetilde{S}_s(t)$ denote the linearly interpolated evaluation success rate in $[0, 1]$ for seed s at environment step t. The normalized AUC, expressed as a percentage, is

$$\mathrm{AUC}_s = \frac{100}{T} \int_0^T \widetilde{S}_s(t)\, \mathrm{d}t. \tag{11}$$

Within each task, all seven conditions and five seeds are compared over the same evaluation interval $[0, T]$, which is covered by every run. For Tasks 1, 3, 7, and 9, the endpoints are 1,175,842, 1,176,236, 1,179,116, and 1,178,331 environment steps, respectively. This metric measures average evaluation success over a shared interaction interval. Last-evaluation success uses each run's last periodic evaluation within the 1.2-million-step training budget. For the four-task average, we first average the task scores within each seed, then compute the mean and sample standard deviation across the five seeds.

Complete RL results. Table 10 reports all seven conditions on the four tasks. Figures 6 and 7 provide the corresponding rollout and evaluation learning curves. Progress + Relation achieves the highest mean evaluation AUC on all four tasks and the highest mean last-evaluation success on all four tasks. It also exceeds Relation-loss-off in mean evaluation AUC on every task, with a four-task average of 36.7% versus 19.1%. This comparison supports a contribution from the relation objective beyond the mixed-sampling recipe. Progress + Relation also exceeds Progress + Preference in mean evaluation AUC (36.7% versus 19.4%), despite the latter's high held-out preference accuracy (Table 11). Relative to Sparse and Robometer, joint supervision achieves higher four-task averages for both metrics. On Task 3, Progress + Relation achieves higher mean evaluation AUC than Robometer (48.9% versus 26.5%) and higher mean last-evaluation success (88.8% versus 70.4%).

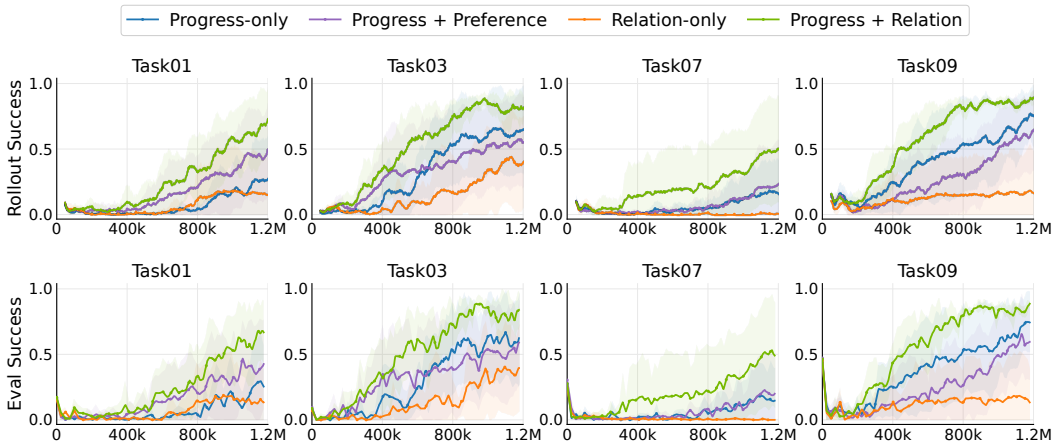


Figure 6: **Complete learning curves for supervision ablations.** Rollout (top) and independent evaluation (bottom) success for four supervision variants on LIBERO-10 Tasks 1, 3, 7, and 9. Lines show means across five environment seeds; shaded bands indicate $\pm 1.96\,\mathrm{SEM}$. The horizontal axis shows environment interaction steps.

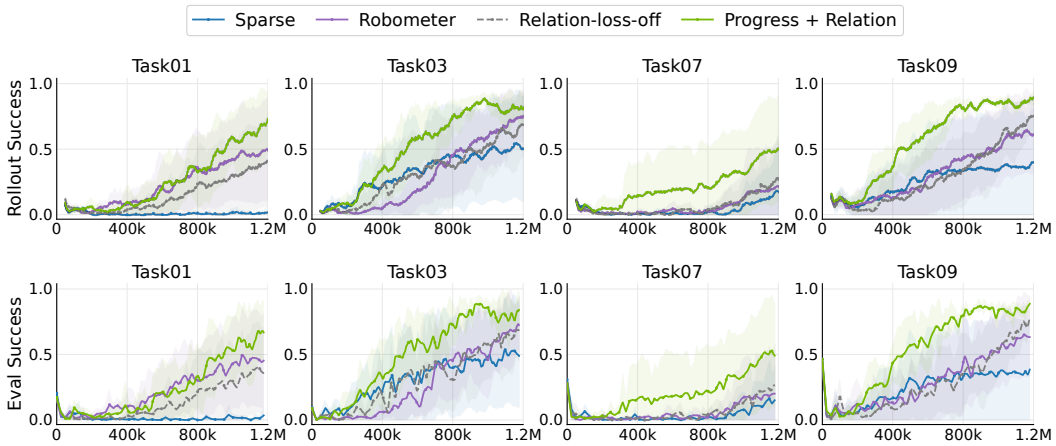


Figure 7: **Additional controls and reward-model baselines.** Progress + Relation is compared with Relation-loss-off, Sparse, and Robometer on the same four LIBERO-10 tasks. Top: rollout success. Bottom: independent evaluation success. Lines show means across five environment seeds; shaded bands indicate $\pm 1.96\,\mathrm{SEM}$. The horizontal axis shows environment interaction steps.

C.1.1 Held-Out Trajectory Preference Prediction

Evaluation protocol. To assess whether Progress + Preference learned the trajectory comparisons used for supervision, we evaluate the trained preference head of the same checkpoint used in the RL ablations, without further training or checkpoint selection. The held-out split contains 46 successful and 93 failed trajectories across all ten LIBERO-10 tasks, with no trajectory overlap with the training or validation splits. We evaluate all 433 unique within-task success–failure pairs, including 73 pairs from the four RL tasks (Tasks 1, 3, 7, and 9). Labels are determined by recorded terminal outcomes. The 103 trajectories used in the progress diagnostics (Appendix C.1.2) form a subset of this split; the present evaluation uses all 139 trajectories because it requires terminal outcomes rather than unambiguous local event annotations. Inputs follow the training protocol: eight uniformly sampled frames per trajectory, together with its start image and a fixed goal image from the training setup. We randomize the primary A/B order with approximately balanced labels within each task (seed 20260923), then evaluate every pair again with the order reversed and its label inverted.

Table 11: Held-out preference prediction on LIBERO-10. S/F denotes successful/failed trajectories. Accuracies are percentages. Brackets give trajectory-bootstrap 95% intervals for primary accuracy; BCE is also measured on the primary order. Aggregate rows weight tasks equally. A degenerate interval at 100% reflects no observed errors, not certainty about population accuracy.

Task	S/F	Pairs	Primary accuracy	Swapped accuracy	BCE
0	6/8	48	100.0 [100.0, 100.0]	93.8	0.0210
1	4/2	8	100.0 [100.0, 100.0]	100.0	0.0187
2	3/8	24	100.0 [100.0, 100.0]	100.0	0.0167
3	5/3	15	93.3 [73.3, 100.0]	100.0	0.0765
4	7/12	84	97.6 [89.3, 100.0]	96.4	0.0663
5	2/5	10	70.0 [20.0, 100.0]	90.0	0.4478
6	7/10	70	92.9 [78.6, 100.0]	92.9	0.2140
7	5/4	20	100.0 [100.0, 100.0]	100.0	0.0138
8	4/31	124	96.0 [87.9, 100.0]	94.4	0.1631
9	3/10	30	100.0 [100.0, 100.0]	100.0	0.0155
All ten (macro)	46/93	433	95.0 [89.9, 99.0]	96.7	0.1053
RL four (macro)	17/19	73	98.3 [93.3, 100.0]	100.0	0.0311

Metrics and uncertainty. Following the preference prediction metric in Robometer (Liang et al., 2026), accuracy is the fraction of comparisons whose predicted ordering matches the ground truth; here, a correct prediction selects the successful trajectory. We additionally report binary cross-entropy (BCE) and order-swap diagnostics. Macro averages weight tasks equally, whereas pooled results weight trajectory pairs equally. We compute percentile 95% intervals using 10,000 bootstrap replicates, independently resampling successful and failed trajectories within each task and weighting each pair by the product of its endpoints' resampling multiplicities. The two orders of a pair remain together. Thus, shared trajectories and mirrored comparisons are not treated as independent observations. The intervals are conditional on the fixed checkpoint and task set and do not capture reward-model training variability. This is a same-task success–failure evaluation, rather than a reproduction of Robometer's out-of-distribution benchmark.

Results and order sensitivity. Table 11 reports per-task and aggregate results. Primary accuracy averaged equally across ten tasks is 95.0% (95% interval: 89.9–99.0%), with BCE 0.1053; accuracy on the four RL tasks is 98.3% (93.3–100.0%). Pooling all 433 pairs gives 96.3% primary accuracy (417/433) and 95.6% after swapping A/B (414/433). The predicted preferred trajectory is unchanged by the swap for 92.8% of pairs. Nevertheless, the model is sensitive to presentation order: across both orders, pooled accuracy is 92.6% when the successful trajectory is in position A and 99.3% when it is in position B. Task 5 has only two successful and five failed trajectories; its primary and swapped accuracies are 70.0% and 90.0%, respectively. We report both orders rather than selecting the more favorable one.

Relation to downstream reward learning. The preference head achieves high held-out accuracy in comparing successful and failed trajectories from the same task. Our RL rewards are shaped using progress predictions at individual time steps, which the preference head's pairwise outputs do not directly provide. Although preference training updates the shared backbone, accurately ranking whole trajectories does not by itself ensure that the progress head provides an effective shaping potential. On the four RL tasks, Progress + Preference achieves a mean evaluation AUC of 19.4%, compared with 36.7% for Progress + Relation (Table 10).

Training diagnostics. As supporting evidence, mean preference accuracy over logged mini-batches increases from 73.9% in updates 1–500 to 97.7% in updates 1501–2000, while mean BCE decreases from 0.434 to 0.069. These summaries use 51 and 50 logged snapshots, respectively; they are training diagnostics rather than evaluations of the entire training set.

Table 12: **Progress prediction diagnostics.** The relation satisfaction rate is averaged equally across nine tasks containing unrecovered regression annotations. Prediction range is averaged across 103 held-out trajectories.

Supervision	Relation satisfaction rate (%)	Mean prediction range
Progress-only	41.62	0.5928
Progress + Preference	48.57	0.5676
Relation-only	58.86	0.0021
Progress + Relation	54.47	0.5097
Relation-loss-off	44.89	0.5816

C.1.2 Progress Prediction Diagnostics

To investigate the poor RL performance of Relation-only, we evaluate predicted progress on 103 held-out trajectories with unambiguous reviewed annotations. This diagnostic set covers all ten LIBERO-10 tasks and includes both successful and failed executions, while the downstream RL ablations use four tasks. We examine whether models correctly predict annotated progress decreases and whether their predictions retain variation across trajectory stages. The latter checks for output compression. We use the same fixed reward-model checkpoints as in the supervision ablations, with identical query frames and goal images across conditions. All diagnostics use the original predictions in $[0, 1]$, without rescaling individual models or trajectories. We also include *Relation-loss-off*, which retains the joint-training sampling recipe but disables the relation loss.

Relation satisfaction rate. We compute this metric on the *unrecovered regression* subset of the 103 held-out trajectories: 46 trajectories spanning nine tasks. These annotations identify achieved progress that is lost and remains unrecovered at the end of the execution. For each trajectory, we compare the prediction at the annotated reference frame with predictions at 16 uniformly spaced frames from 1.5 seconds after the reference to the end of the trajectory. A comparison satisfies the relation when the event-frame prediction is strictly lower than the reference prediction; ties count as unsatisfied comparisons. We average the relation satisfaction rate within each trajectory, then within each task, and finally equally across the nine tasks.

Within-trajectory prediction range. To examine whether predictions distinguish different trajectory stages, we sample nine evenly spaced frames from the beginning to the end of each of the 103 trajectories. For trajectory i, we compute

$$\Delta_i = \max_{t \in S_i} p_{i,t} - \min_{t \in S_i} p_{i,t}, \qquad |S_i| = 9,$$

where $p_{i,t}$ is predicted progress and S_i is the sampled frame set. We report the unweighted mean of Δ_i across trajectories. This measure diagnoses output compression; a larger range alone does not establish more accurate progress predictions.

Results and interpretation. Table 12 shows that Relation-only achieves the highest relation satisfaction rate among these variants, yet its mean prediction range is only 0.0021, compared with 0.5097 for joint supervision. Across all queried frames in the held-out set, its predictions lie between approximately 0.0010 and 0.0051. Thus, some relative orderings remain correct even though different trajectory stages receive nearly identical, near-zero scores. Under the fixed shaping coefficient used in our RL experiments, predictions at this scale would yield small progress-dependent terms in the chunk-level reward formulation (Equation 10). These held-out observations are therefore consistent with a weakened shaping signal. Joint supervision improves the relation satisfaction rate over Progress-only while retaining substantial prediction variation. Together with the RL results in Section 4.1, these diagnostics support the complementary roles of temporal labels in anchoring the numerical scale of predicted progress and progress relations in constraining local progress changes.

An example of progress-scale collapse. Figure 8 compares the three reward models on the same held-out failure trajectory from LIBERO-10 Task 7. Relation-only remains close to zero at all sampled points, with predictions ranging from 0.0013 to 0.0036. For decrease-relation evaluation, all

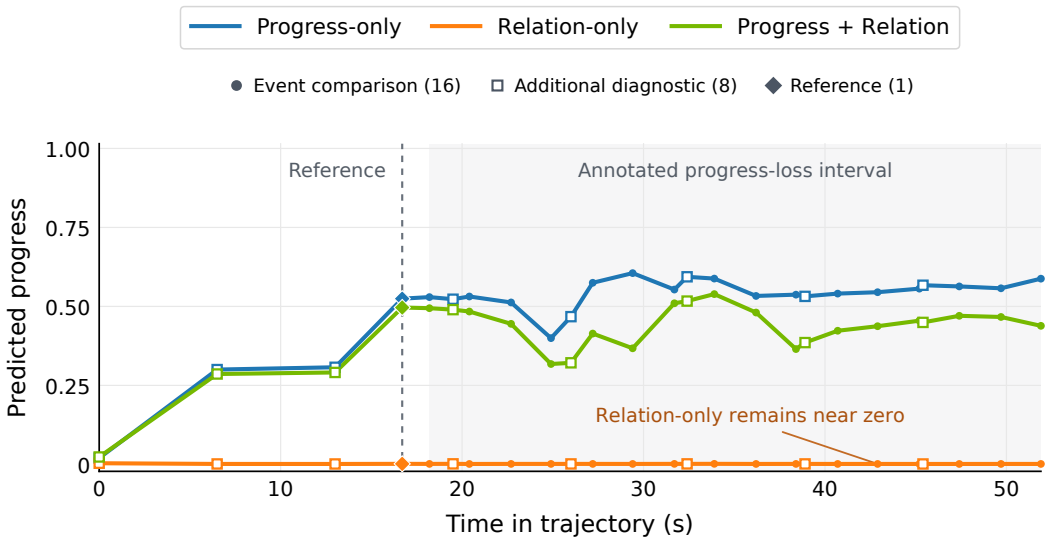


Figure 8: **A held-out example of progress-scale collapse.** Predicted progress on a LIBERO-10 Task 7 trajectory, shown on a common, unnormalized scale. Each curve connects 25 sampled predictions with straight lines: 16 event-comparison points (filled circles), eight additional trajectory-diagnostic points (open squares), and one reference point (diamond). The dashed line marks the reference time, and shading indicates the progress-loss comparison interval. Only filled circles enter the reported decrease counts; the five open squares within the shaded interval are excluded.

three models use the same 16 uniformly sampled comparison frames (filled circles). Each model's predictions at these frames are compared with its own prediction at the event reference. Relation-only assigns lower progress to 15 of these 16 frames. Progress-only retains substantial score variation but satisfies only 2 of the same 16 comparisons; joint supervision preserves score variation while satisfying 14. This example shows that correct decrease-relation judgments can coexist with a severely compressed predicted progress scale.

Table 13: Reward-model adaptation settings for LIBERO. Learning rates are base values before scheduling. Event2Reward batches count flattened input clips (model queries), whereas Robometer batches count trajectory pairs. Global batches include all devices and gradient accumulation.

Setting	Event2Reward	Robometer
Optimizer updates	500	500
Optimizer	AdamW	AdamW
Language learning rate	2×10^{-6}	2×10^{-5}
Vision learning rate	5×10^{-7}	2×10^{-5}
Prediction-head learning rate	10^{-5}	2×10^{-5}
Trainable language parameters	All	All
Trainable vision modules	Last 3 blocks and merger	Entire encoder
Trainable prediction heads	Progress, trend	Progress, preference, success
Per-device microbatch	10 clips	8 pairs
Devices	2	2
Gradient accumulation	4	1
Global batch per update	80 clips	16 pairs
Warmup updates	25	50
Schedule after warmup	Cosine	Cosine
Weight decay	0.01	0.01

C.2 Policy Learning in Simulation

C.2.1 Reward-Model Adaptation

We adapt reward models for four LIBERO suites: Spatial, Object, Goal, and LIBERO-10, each containing ten tasks. Event2Reward starts from the model obtained after joint training described in Section 3.3. Robometer (Liang et al., 2026) and Robo-Dopamine (Tan et al., 2025) start from their publicly released pretrained checkpoints.

Within each suite, Event2Reward and Robometer use the same 200 adaptation trajectories. The Spatial set contains 114 expert demonstrations, 58 suboptimal successful rollouts, and 28 failed rollouts; the Object, Goal, and LIBERO-10 sets each contain 75, 64, and 61 trajectories of these categories, respectively. Suboptimal and failed rollouts are collected using `pi0_libero` and manually annotated and checked. Each method follows its own adaptation recipe consistently across suites for 500 updates, and the final checkpoint is used for policy learning. Event2Reward uses temporal labels and auxiliary trend supervision on expert demonstrations, together with progress relations derived from annotated events in suboptimal and failed executions. Robometer retains its progress, preference, and success objectives. Table 13 summarizes the optimization settings used consistently across the four suites.

Robo-Dopamine uses task-specific one-shot adaptation. For each task, we select one expert demonstration from the official LIBERO data and adapt the pretrained model for two epochs using its released training objective. The final frame of the demonstration serves as the goal image. This produces one reward model per task, whereas Event2Reward and Robometer each use one adapted model per suite. Sparse uses only the environment reward. All reward models remain fixed during policy learning, and each adapted model is reused across environment seeds.

C.2.2 Policy Learning and Evaluation

For each suite, we select one expert demonstration from each of its ten tasks and jointly fine-tune $\pi_{0.5}$ (Physical Intelligence et al., 2025) on these demonstrations. Within each task, all reward conditions share the same frozen base policy. Reinforcement learning trains task-specific residual policies, allowing us to compare how different rewards support improvement from the same policy initialization.

We use the residual SAC framework described in Appendix C.1, with action-chunk length $H = 5$. LIBERO-Goal and LIBERO-Object use residual scale $\alpha = 0.2$ and discount $\gamma = 0.998$; LIBERO-

Spatial uses $\alpha = 0.5$ and $\gamma = 0.99$; LIBERO-10 uses $\alpha = 0.1$ and $\gamma = 0.998$. Reward construction and the remaining SAC settings follow that framework. Reinforcement learning updates the residual actor, critics, and temperature parameter using online experience. Reward-model adaptation trajectories are not inserted into the policy replay buffer.

For online reward inference, Event2Reward uses the task instruction, the episode's initial main-camera image, a fixed goal image from the same view, and the latest five consecutive main-camera observations. Robometer uses the instruction and up to eight main-camera frames sampled from the observed trajectory prefix, taking the current-frame progress clipped to $[0, 1]$. Robo-Dopamine uses forward inference with fixed demonstration reference and goal images, together with main- and wrist-camera observations at the episode start and current step. Its percentage score is divided by 100 and clipped to $[0, 1]$.

The environment provides a sparse reward of 1 upon success and 0 otherwise, with no additional failure penalty. Episodes terminate upon success or are truncated after 220, 280, 300, and 520 environment steps for Spatial, Object, Goal, and LIBERO-10, respectively, in both training and independent evaluation.

Each method is evaluated with five environment seeds per task: 7, 8, 9, 17, and 27. Quantitative comparisons use budgets of 200k training environment steps per run for Spatial, Object, and Goal, and 1.2M for LIBERO-10. For Spatial, Object, and Goal, learning curves are displayed through 240k steps, while evaluations beyond 200k steps are excluded from the quantitative statistics. After every 50 completed training episodes, a frozen policy snapshot is evaluated for 50 episodes in a separate environment. Evaluation uses deterministic residual actions while retaining the base policy's default sampling. Evaluation interactions are excluded from the training budget and are not added to training replay.

Metrics. Policy success is taken from the last periodic independent evaluation within the corresponding suite's training budget. Since evaluations are triggered by completed episodes, this evaluation need not occur exactly at the budget endpoint. We first average success rates over the ten tasks within each seed, then report the mean and sample standard deviation across seeds.

Training rollout curves. Rollout success rates summarize the training episodes collected by the evolving policy. We retain the episode windows used in the original logs. On LIBERO-Goal, Sparse and Event2Reward use the most recent up to 100 logged episodes, including the available episodes before the window is full; Robo-Dopamine and Robometer use the most recent 100 completed episodes and begin logging after 100 completions. LIBERO-Object uses the most recent 100 completed episodes for all methods. LIBERO-Spatial uses the most recent 100 logged episodes for all methods and begins logging once this window is full; its source logs may include a budget-truncated episode at the end of training. We apply no additional temporal smoothing to these exported values. Rollout curves describe training behavior; policy-performance, AUC, and crossing-time comparisons use independent Eval.

Normalized evaluation AUC. For each run, we compute normalized evaluation AUC as in Equation 11 using the corresponding interaction budget: $T = 200{,}000$ for Spatial, Object, and Goal, and $T = 1{,}200{,}000$ for LIBERO-10. We use the recorded step-zero base-policy evaluation and linearly interpolate between subsequent evaluations within the comparison budget. The last observed success rate is held constant from its evaluation step to T, so no evaluation beyond the budget contributes to the integral. We first average the ten task AUCs within each seed, then report the mean and sample standard deviation across the five seeds in Table 14.

Sample efficiency. We measure the training interactions needed to first reach a target suite-average success rate. At each recorded evaluation step, we use the most recent available evaluation for every run, including each run's own step-zero evaluation before its first subsequent evaluation. We average the ten task scores within each seed and then average the five seed means. The first step at which this suite-average curve reaches or exceeds the target is the reported crossing time. This is a crossing of the aggregate curve, not an average of per-task or per-run crossing times. Table 15 reports targets of 50%, 60%, and 70%.

Table 14: **Normalized evaluation AUC on LIBERO.** AUC is normalized by the training budget: 200k environment steps for Spatial, Object, and Goal, and 1.2M for LIBERO-10. Values (%) are reported as mean ± sample standard deviation across five environment seeds, after averaging the ten task AUCs within each seed. Robometer denotes the adapted model. Bold indicates the highest mean in each suite.

Method	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-10
Sparse	22.44 ± 3.67	20.90 ± 1.21	40.10 ± 1.65	15.61 ± 5.18
Robo-Dopamine	22.44 ± 1.67	24.42 ± 1.94	37.20 ± 1.78	16.21 ± 3.34
Robometer	20.05 ± 2.77	24.83 ± 3.63	33.72 ± 2.49	17.32 ± 6.04
Event2Reward	$\mathbf{25.55} \pm 1.66$	$\mathbf{33.06} \pm 2.28$	$\mathbf{44.91} \pm 1.51$	$\mathbf{24.91} \pm 4.31$

Table 15: **Environment interactions to first reach a target suite-average success rate.** The comparison budget is 200k environment steps for Spatial, Object, and Goal, and 1.2M for LIBERO-10. Each suite-average curve averages the ten tasks within each environment seed and then averages the five seed means. A dash denotes no recorded crossing within the budget. Bold indicates the earliest crossing in each suite at each target.

Method	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-10
(a) Target: 50% mean success				
Sparse	–	–	107,329	–
Robo-Dopamine	–	173,777	126,216	–
Robometer	–	177,230	162,312	–
Event2Reward	–	**141,119**	**102,873**	**1,149,903**
(b) Target: 60% mean success				
Sparse	–	–	162,154	–
Robo-Dopamine	–	–	156,630	–
Robometer	–	199,163	–	–
Event2Reward	–	**160,052**	**132,806**	–
(c) Target: 70% mean success				
Sparse	–	–	–	–
Robo-Dopamine	–	–	–	–
Robometer	–	–	–	–
Event2Reward	–	**178,441**	**164,080**	–

For each target and suite, we compare Event2Reward with the baseline that reaches the target in the fewest interactions. If both have a recorded crossing, the percentage reduction is $100(1 - \tau_{\text{Event2Reward}}/\tau_{\text{baseline}})$, using unrounded crossing steps. At 60% success, these steps are 132,806 versus 156,630 for Event2Reward and Robo-Dopamine on LIBERO-Goal, and 160,052 versus 199,163 for Event2Reward and Robometer on LIBERO-Object, giving reductions of 15.2% and 19.6%. If a method has no recorded crossing within the comparison budget, we mark the entry as unreached and do not substitute the budget endpoint as a crossing time.

C.2.3 Complete Task Results

Figures 9, 11, 13, and 15 show independent-evaluation (Eval) learning curves for LIBERO-Goal, LIBERO-Object, LIBERO-Spatial, and LIBERO-10, respectively. Figures 10, 12, 14, and 16 show the corresponding training-rollout (Rollout) curves. The two views are paired by suite, with Eval followed by Rollout. Each figure includes all ten tasks in the original suite ordering: Task00–Task04 in the upper row and Task05–Task09 in the lower row.

For LIBERO-Goal, LIBERO-Object, and LIBERO-Spatial, both Eval and Rollout curves are displayed through 240k training environment steps; quantitative comparisons retain the 200k-step budget. Lines show means across five environment seeds, with shaded bands indicating $\pm 1.96\,\mathrm{SEM}$, clipped to $[0, 1]$. Seed curves are aligned by linear interpolation on a shared step grid containing their recorded observation steps. No additional smoothing or extrapolation is applied; endpoint values are interpolated only between recorded observations. At step zero, the reference shows the performance of the shared frozen base policy with zero residual actions. At positive training steps, Eval curves report independent policy evaluations, while Rollout curves summarize success over logged training episodes. AUC and target-crossing statistics follow the evaluation protocol in Appendix C.2.2.

The LIBERO-10 figures show the longer learning trajectories within the 1.2M-step training budget. Each task uses its fixed common observation interval, ending at the last supported step in that interval without extrapolation to the nominal budget endpoint. This plotting endpoint differs from the per-run last periodic evaluation used in Table 1. Step zero in the Eval curves denotes evaluation of the frozen base policy with the residual action set to zero. Rollout values use the most recent 100 logged training episodes and are displayed from the first common point with a full window, without adding an Eval reference at step zero. The same five-seed averaging and uncertainty bands apply.

On some tasks, success initially drops when residual learning begins. This observation is consistent with the early exploration behavior discussed in PLD (Xiao et al., 2026), where the residual policy starts to deviate from the base policy and visits potentially suboptimal states.

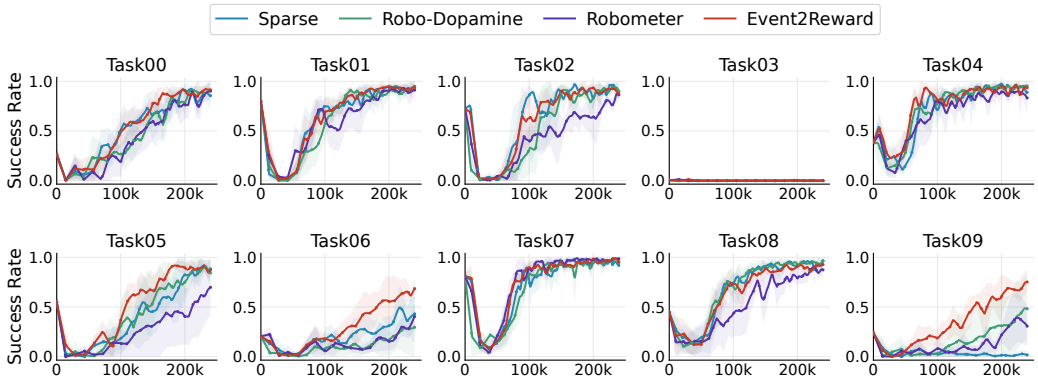


Figure 9: **Independent evaluation on all ten LIBERO-Goal tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps. Curves extend to 240k steps; quantitative comparisons use 200k. Lines and shaded bands show the mean and ± 1.96 SEM across five environment seeds.

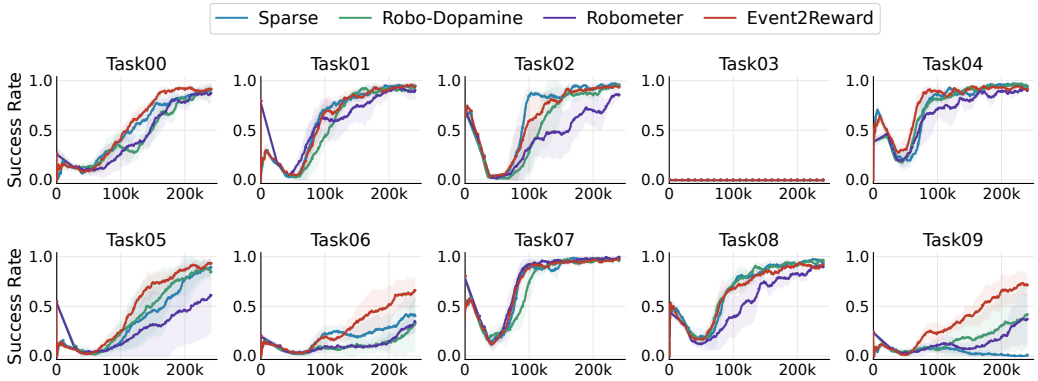


Figure 10: **Training rollout success on all ten LIBERO-Goal tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps; curves extend to 240k steps. Lines and shaded bands show the mean and ± 1.96 SEM across five environment seeds, using the logged episode windows described above. Step zero indicates the shared base-policy evaluation. Quantitative comparisons use independent Eval within 200k steps.

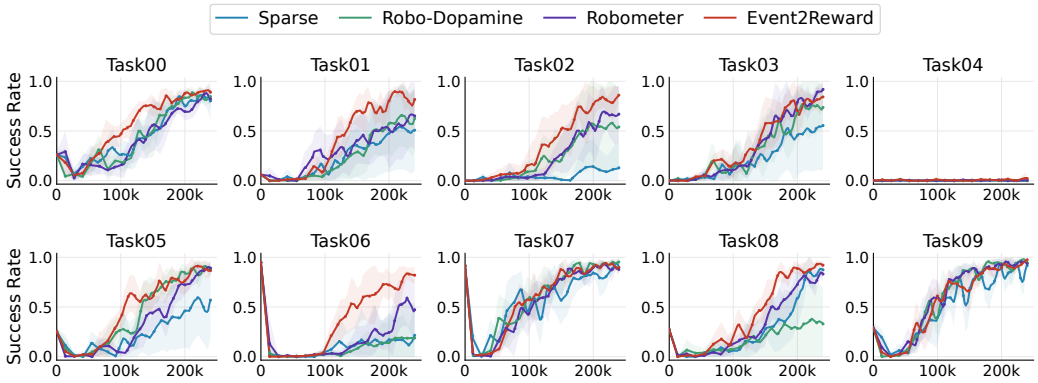


Figure 11: **Independent evaluation on all ten LIBERO-Object tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps. Curves extend to 240k steps; quantitative comparisons use 200k. Lines and shaded bands show the mean and $\pm 1.96\,\mathrm{SEM}$ across five environment seeds.

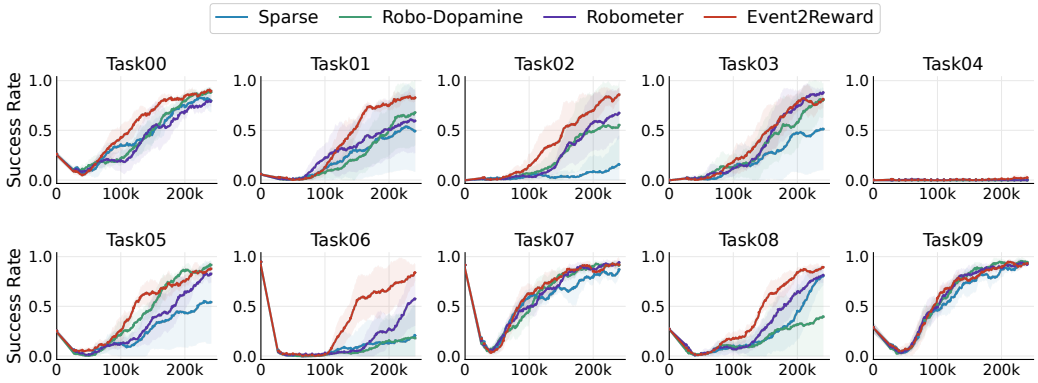


Figure 12: **Training rollout success on all ten LIBERO-Object tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps; curves extend to 240k steps. Lines and shaded bands show the mean and $\pm 1.96\,\mathrm{SEM}$ across five environment seeds, using the logged episode windows described above. Step zero indicates the shared base-policy evaluation. Quantitative comparisons use independent Eval within 200k steps.

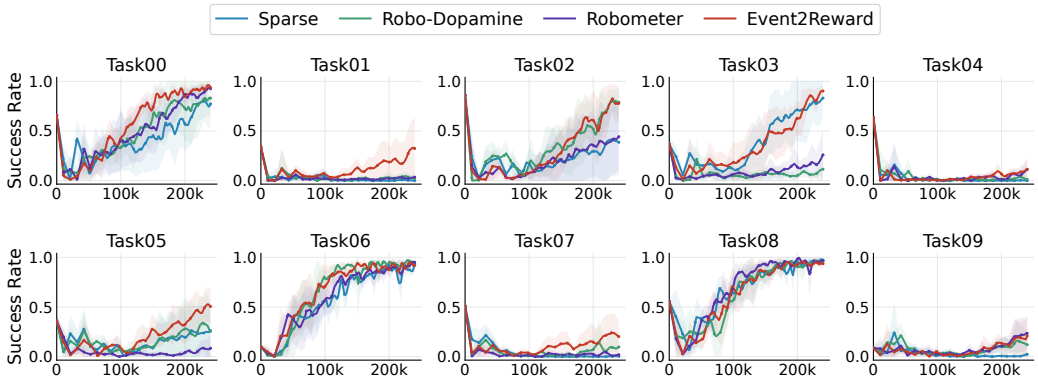


Figure 13: **Independent evaluation on all ten LIBERO-Spatial tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps. Curves extend to 240k steps; quantitative comparisons use 200k. Lines and shaded bands show the mean and ± 1.96 SEM across five environment seeds.

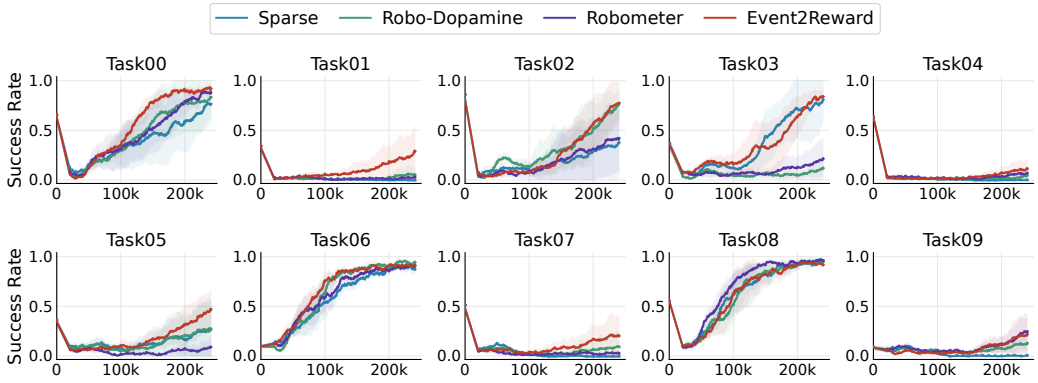


Figure 14: **Training rollout success on all ten LIBERO-Spatial tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps; curves extend to 240k steps. Lines and shaded bands show the mean and ± 1.96 SEM across five environment seeds, using the logged episode windows described above. Step zero indicates the shared base-policy evaluation. Quantitative comparisons use independent Eval within 200k steps.

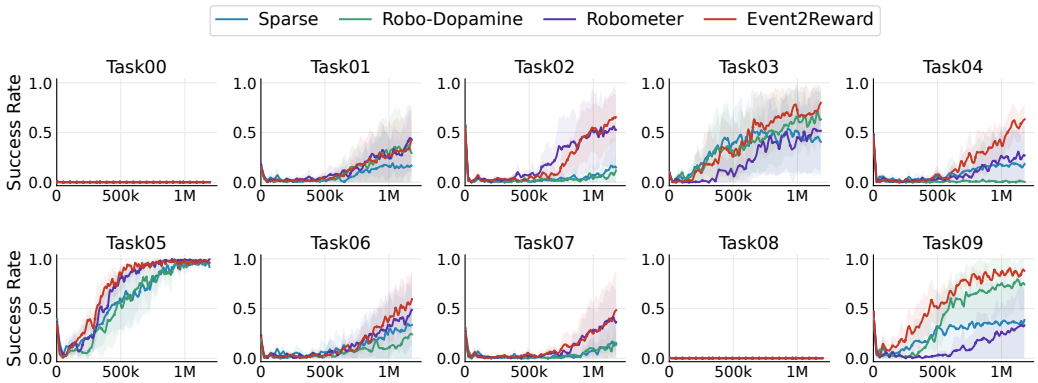


Figure 15: **Independent evaluation on all ten LIBERO-10 tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps within the 1.2M-step budget; each task is plotted over its fixed common observation interval without extrapolation. Lines and shaded bands show the mean and ± 1.96 SEM across five environment seeds.

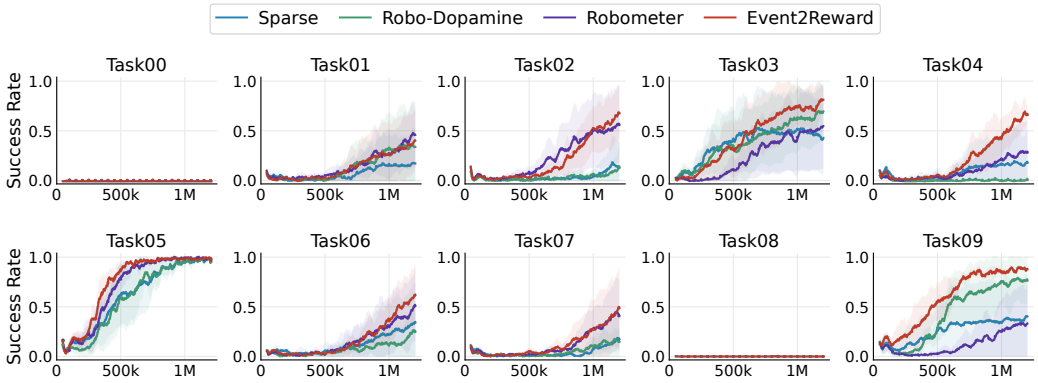


Figure 16: **Training rollout success on all ten LIBERO-10 tasks.** Task IDs follow the original suite ordering. The horizontal axis shows training environment steps within the 1.2M-step budget. Lines and shaded bands show the mean and ± 1.96 SEM across five environment seeds, using the most recent 100 logged training episodes. Curves begin once all runs have a full window and remain within the fixed common observation interval; no step-zero Eval reference is added.

C.2.4 ROLLOUT JOINT-TRAINING ABLATION

Training controls. Event2Reward w/o rollout joint training uses the 50,000-update pretraining checkpoint; Event2Reward uses the 10,000-update joint-training checkpoint initialized from it. Both share Qwen3.5-4B and the target-adaptation objective. All conditions use 500 adaptation updates, 80 queries per update, and training seed 42. With 200 trajectories, the ablated model uses eight devices and one accumulation step; all other conditions use two devices and four accumulation steps, preserving the global batch. Joint training combines real rollouts, 5,452 simulated RoboFAC rollouts, and pretraining-data replay, adding 10,000 updates (800,000 queries; Appendix B.4). The comparison tests this stage as a whole, without separately controlling data source, scale, supervision, or additional compute.

Fixed adaptation subset. The 40-trajectory subset contains 15 normal successful trajectories and 25 event-bearing trajectories, sampled from the same 200-trajectory Object adaptation set used in Section 4.2. Normal successes cover all ten tasks, with at least one per task. The four event categories retain 7, 5, 11, and 2 trajectories for no further progress, unrecovered regression, progress stall, and recoverable regression, respectively. We allocate event counts by their original proportions using the largest-remainder rule, then stratify by task, with ascending task IDs breaking allocation ties. Within each stratum, a fixed hash ordering using sampling seed 20260925 determines the trajectories. The subset manifest is frozen before training without consulting model predictions or RL outcomes. All variants share ten fixed goal images; nine are sourced outside the 40-trajectory subset. The reduced-data condition therefore uses 40 adaptation trajectories plus these shared goal images.

RL and evaluation. Each condition has one reward-model checkpoint and 50 RL runs: ten Object tasks with environment seeds $\{7, 8, 9, 17, 27\}$. All share the frozen Object one-shot $\pi_{0.5}$ checkpoint at step 49,999 and residual SAC settings from Appendix C.2: residual scale 0.2, discount 0.998, and chunk horizon five. Runs train for 300k environment steps; Table 2 retains the 200k comparison budget. Every 50 completed training episodes, independent evaluation uses 50 episodes with deterministic residual and sampled base-policy actions. Success is the last periodic evaluation within budget, not an evaluation at a common endpoint checkpoint. AUC uses each run's initial evaluation, linearly interpolates between in-budget observations, and holds the last value to the budget endpoint; no out-of-budget observation contributes.

Uncertainty and scope. Pairing the task-averaged results by environment seed, the 200k AUC gain from rollout joint training is $+3.39$ percentage points with 200 adaptation trajectories, with a nominal 95% paired t interval of $[-2.46, 9.24]$; with 40 trajectories, it is $+4.13$ points with interval $[-0.63, 8.89]$ (four degrees of freedom). These intervals reflect environment-seed variation conditional on one reward-model training seed and one fixed 40-trajectory subset. The positive mean differences do not establish statistical significance or equivalent performance after reducing adaptation data by 80%. Both methods already include the same pretraining and relation-supervised target adaptation; this comparison addresses the incremental rollout joint-training stage.

Full training budget. Table 16 reports the same metrics at the actual 300k-step training budget. Rollout joint training increases mean AUC by 3.23 points with 200 trajectories and 4.77 points with 40 trajectories; last periodic evaluation success increases by 4.44 and 6.20 points, respectively. The final periodic evaluations occur between 290,617 and 300,000 environment steps. Benefits vary across tasks: Task02 and Task06 account for 5.68 points of the 6.20-point mean success gain in the 40-trajectory comparison. These results complement the fixed 200k comparison rather than changing its reporting budget.

Table 16: **Rollout joint-training ablation at the full 300k budget.** Evaluation AUC and last periodic evaluation success (%) use the same task and seed aggregation as Table 2. Values are mean ± sample standard deviation across five environment seeds.

Method	Adaptation trajectories	Eval AUC	Success
Event2Reward w/o rollout joint training	200	45.49 ± 2.25	80.20 ± 0.62
Event2Reward	200	48.72 ± 1.60	84.64 ± 1.28
Event2Reward w/o rollout joint training	40	42.28 ± 4.00	77.72 ± 9.06
Event2Reward	40	47.05 ± 2.54	83.92 ± 2.06

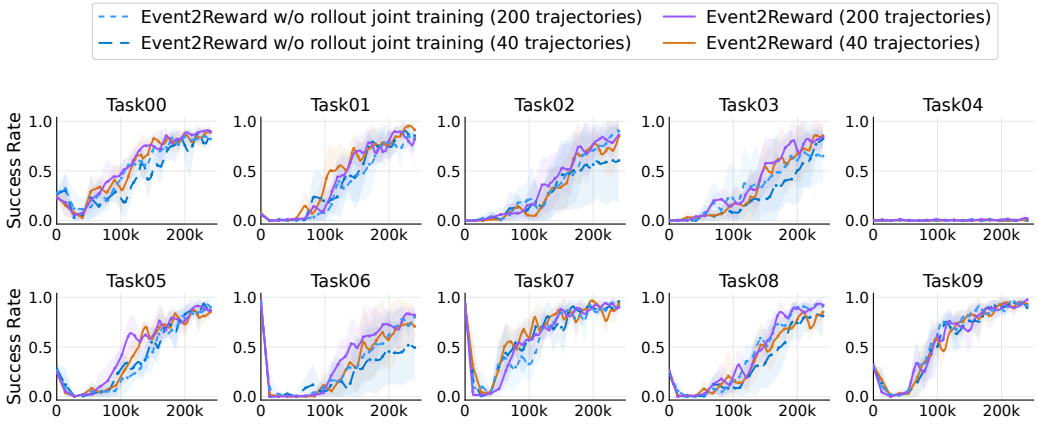


Figure 17: **Rollout joint-training ablation: independent evaluation.** Each task compares Event2Reward with Event2Reward w/o rollout joint training using 200 or 40 adaptation trajectories. Lines show means across five environment seeds and bands show $\pm 1.96\,\mathrm{SEM}$, clipped to $[0, 1]$. Curves extend to 240k training environment steps; the main quantitative comparison uses 200k. Curves are linearly interpolated only within the current runs' common observed interval for each task, with no additional smoothing or extrapolation.

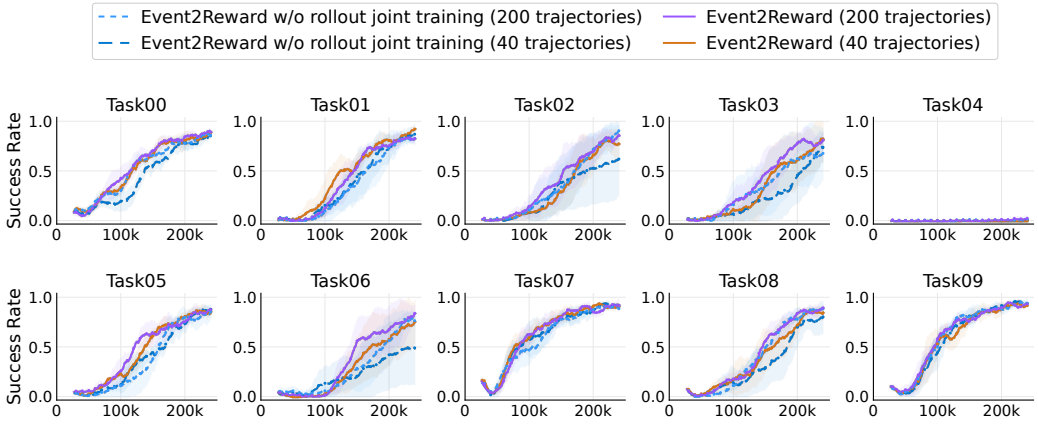


Figure 18: **Rollout joint-training ablation: training rollout success.** The four variants and seed aggregation match Figure 17. Each run uses its logged 100-completed-episode success window, without additional smoothing. Curves begin once all runs in a task have a full window and stay within the current runs' common observed interval through 240k environment steps. No step-zero evaluation is inserted into the training-rollout curves.

D Real-World Tasks and Evaluation Protocols

D.1 Insertion Tasks

The three SERL tasks use the insertion setups shown in Figure 4 (top). Insertion success is judged by a human operator according to the task-specific criteria below.

Plug Insert. The robot starts with the plug already held in its gripper, aligns it with the target socket, and inserts it fully. An episode succeeds when the plug is fully seated in the socket. The maximum episode length is 190 steps.

RAM Insert. The robot starts with the RAM module already held in its gripper, aligns it with the target memory slot, and presses it into place. An episode succeeds when the module is fully inserted and seated; engagement of the retaining clips is not required. The task has an insertion tolerance of approximately 2 mm. The maximum episode length is 400 steps.

Ethernet Insert. The connector initially rests on the tabletop. The robot must grasp it, lift the connector and the attached cable, and insert it into the target Ethernet port. An episode succeeds when the connector is fully seated in the port. The maximum episode length is 300 steps.

D.2 Object-Manipulation Tasks

The three DSRL tasks use a π_0 base policy on Franka robots. Success is assessed at the episode level according to the task-specific criteria below.

Stack Bowls. Three bowls are placed on the tabletop, with one bowl already at the central target position. The robot must place the other two bowls into the central bowl to form a stack of three. An episode succeeds only when all three bowls form a stable stack after the gripper releases them. The maximum episode length is 800 environment steps.

Press Buttons. Blue, pink, and yellow buttons are placed on the tabletop. The robot must press them downward in the order blue, pink, then yellow. An episode is successful only when all three buttons are pressed in this order. The maximum episode length is 500 environment steps.

Plate Vegetables. The robot must pick up a cabbage, an ear of corn, and a pepper from the tabletop and place all three on the target plate. An episode succeeds only when all three objects remain stably on the plate after the gripper releases them. The maximum episode length is 800 environment steps.

Figure 4 (bottom) illustrates these tasks. Their quantitative results are reported alongside the insertion results in Figure 5.

D.3 Training and Evaluation Protocols

SERL offline demonstrations. For each of the three SERL insertion tasks, we populate the offline demonstration buffer with 20 demonstrations of that task. Within the same task, Sparse, Event2Reward, Robometer, and Robo-Dopamine reuse this demonstration set for policy learning. We do not perform behavior-cloning pretraining.

SERL initialization and autonomous rollouts. For each independent SERL run, we initialize the online replay buffer with 10 human-assisted trajectories collected under the same protocol across reward conditions. These trajectories are collected anew for each run. All subsequent training rollouts and evaluation episodes are executed autonomously. Initialization time is included in the reported training budget, while initialization trajectories are excluded from rollout-success statistics.

DSRL initialization and online learning. For each task, we use the 20 policy demonstrations to fine-tune π_0 with LoRA. All reward conditions for that task start from the same fine-tuned checkpoint. During online reinforcement learning, π_0 remains frozen and only the latent-noise control

policy is trained. DSRL uses action chunks of 25 environment steps. Online exploration and evaluation are autonomous.

Reward-model adaptation. Robo-Dopamine follows its officially recommended one-shot adaptation protocol. For each task, Robometer and Event2Reward are adapted using the same 10 trajectories: five successful and five suboptimal executions. Event annotations used by Event2Reward on these real-robot trajectories follow the procedure in Appendix A.3. For Event2Reward, we annotate the suboptimal trajectories with event types and temporal anchors—reference states, event intervals, and recovery states where applicable—without assigning framewise absolute progress scores. Annotating the five suboptimal trajectories took only one minute of human effort per task. Both methods use 500 adaptation updates and retain their respective simulation-adaptation learning rates, batch sizes, and trainable parameter groups (Appendix C.2.1). The final adapted checkpoint is used for policy learning. All reward models remain frozen throughout online policy learning.

Reward annotation and computation. The sparse task reward is 1 upon human-judged task success and 0 otherwise, with no additional failure penalty. Learned-reward conditions use potential-based reward shaping (PBRS), with predicted progress as the potential, following Equation 2. Shaping rewards are computed at action-chunk boundaries with $\beta = 1$ and $\gamma = 0.999$. The potential at absorbing terminal states is set to zero. For SERL with learned rewards, rewards for offline demonstration trajectories are precomputed with the corresponding reward model before the transitions enter the offline buffer. During online collection, the actor completes an episode without waiting for reward inference. Rewards for the collected trajectory are computed during the subsequent reset, and the transitions enter the replay buffer once their rewards are available.

Reset randomization and episode termination. For each of the three insertion tasks, the starting position is randomized at every reset with independent perturbations along the X, Y, and Z axes, with a perturbation magnitude of approximately 1 cm per axis. An episode ends when the task succeeds, failure is declared, or the applicable step limit is reached. Once failure is declared, the episode ends and the environment is reset; no further attempt is allowed within that episode. During SERL training, a human operator may terminate an episode early if the motion deviates substantially in a potentially unsafe direction. This ends the episode rather than providing corrective actions to the policy.

Training budget and independent evaluation. The main comparison uses 30 minutes of online wall-clock training per run, including initial buffer warm-up where applicable, environment resets, and reward computation. Each reward condition is trained independently three times per task. We evaluate each run's checkpoint at 30 minutes over 20 autonomous episodes, yielding 60 evaluation episodes per task and reward condition across the three trained policies. We compute the success rate for each trained policy, then report the mean and sample standard deviation across the three runs. Setting-level means weight the three tasks equally and are computed before rounding. Human-aborted episodes count as failures and remain in the success-rate denominator.

Plug-insertion training-time comparison. We conduct one additional plug-insertion training run for each of Event2Reward and Sparse under the same SERL framework. For each run, we report the elapsed wall-clock time until a rolling window of the 10 most recent autonomous training episodes first contains only successful episodes: 15 minutes for Event2Reward and 45 minutes for Sparse. The window includes only autonomous episodes after the initial buffer warm-up. Warm-up trajectories are excluded from the window, while their collection time and environment resets are included in both reported times. This training-rollout metric is distinct from the independent evaluation after 30 minutes of training across three runs per condition in Figure 5.