CO-EVOLVING AGENTS: LEARNING FROM FAILURES AS HARD NEGATIVES

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving agents improve their performance on long-horizon tasks by learning from their own interactions with an environment. A common approach uses the resulting failed trajectories as negatives for preference training. However, collecting an agent's own failures does not ensure that they provide informative supervision for further improvement. Obvious failures may be easy to reject without learning to identify errors in more plausible attempts. These plausible but incorrect trajectories can serve as *hard negatives*, helping agents learn to distinguish successful behavior from failures with less obvious errors. To enable agents to generate and learn from such informative negatives, we propose a co-evolving framework that couples a target agent's self-improvement with a failure agent trained to generate hard negatives. The failure agent learns exclusively from both agents' failed trajectories to generate higher-reward failures, which the target agent uses as negatives for preference training. As the target policy improves, its failures provide new training data for the failure agent. The updated failure agent, in turn, provides new negatives for target training, allowing negative generation to adapt alongside the target policy. Across online shopping, scientific reasoning, and interactive SQL querying, our framework improves average task reward by 5.7 over the baseline. These results demonstrate the value of learning to generate and use informative failures for agent self-improvement. We will make our code publicly available.

1 INTRODUCTION

Self-evolving agents improve their performance on long-horizon tasks by learning from their own interactions with an environment. Agents built on large language models (LLMs) carry out tasks such as online shopping, scientific reasoning, and interactive SQL querying by taking actions and observing the results (Yao et al., 2022; Wang et al., 2022; Yang et al., 2023). Completing these tasks requires a sequence of decisions, with each action informed by observations from earlier steps. Starting from a policy trained on expert demonstrations (Chen et al., 2023; Zeng et al., 2024), they collect trajectories of these interactions and use them to update the policy (Song et al., 2024; Yuan et al., 2025; Qi et al., 2025). Repeating this process reduces the need for additional human supervision, which can be costly and time-consuming. Continued improvement depends on whether the self-generated trajectories provide useful supervision for the next round of training.

Recent studies use failed trajectories from these interactions as additional training data. Some construct trajectories that show how an agent can recover from an error and complete the task, then use them for iterative self-training (Yuan et al., 2025). Others train the agent to prefer expert demonstrations over its own failures (Song et al., 2024). Using the agent's own failures allows preference training to address mistakes made by its current policy. However, these negatives are collected from a policy trained to solve tasks, rather than one trained to generate informative failures. Some failures contain obvious errors and are easy to distinguish from expert demonstrations. Learning to reject them may provide limited supervision for recognizing errors in more plausible attempts. Additional sampling increases the number of candidates, but does not directly train the policy to generate informative negatives.

Prior work in contrastive learning and preference optimization has demonstrated the benefits of selecting *hard negatives* that are difficult to distinguish from positive or preferred examples (Robinson et al., 2021; Chen et al., 2026). For agent training, this motivates using failures that require a closer

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.

comparison with successful behavior to identify what went wrong. Consider a shopping task that specifies a product type, color, and budget. Selecting an unrelated product is an obvious failure. Selecting the requested type and color while exceeding the budget satisfies part of the instruction but still fails the task. Comparing this attempt with a successful purchase highlights the missed budget constraint. As the target policy improves, failures that were once difficult to distinguish may become easy to reject. Repeated training on these familiar failure patterns risks overfitting to the observed negatives rather than learning to recognize errors in new trajectories. This motivates training a negative generator on the changing experience of the target policy.

To integrate negative generation with agent self-improvement, we propose a co-evolving framework that trains a failure agent to generate hard negatives and a target agent to learn from them (Figure 1). The failure agent learns exclusively from both agents' failed trajectories, with the objective of generating high-reward failures. We use environment reward, which reflects partial task completion, as a proxy for negative difficulty. The target agent learns to prefer expert demonstrations over failures generated by either agent. Training a separate failure agent therefore adds a source of negatives whose generation is explicitly optimized for target training. This makes learning to generate hard negatives part of the agent's self-improvement process.

The two agents co-evolve through an exchange of self-generated trajectories. After each round of training, the target agent collects new trajectories with its updated policy. Its failures provide new training data for the failure agent. In turn, trajectories from the updated failure policy provide negatives for subsequent target training. Thus, target training changes the experience available to the failure agent, while failure-agent training changes the negatives available to the target agent. Preference labels come from existing demonstrations and environment rewards, so no additional human preference annotations are required. The failure agent adds trajectory collection and optimization costs during training; only the target agent is used at inference time.

We evaluate our framework on WebShop for online shopping (Yao et al., 2022), ScienceWorld for scientific reasoning (Wang et al., 2022), and InterCodeSQL for interactive SQL querying (Yang et al., 2023). Across these tasks, our framework consistently achieves higher task rewards than the baseline, with an average improvement of 5.7. These improvements suggest that learning to generate hard negatives provides useful supervision beyond the target agent's own failures and supports agent self-improvement across domains.

Our contributions are threefold:

- We introduce a failure agent that learns from failed trajectories to generate high-reward failures as hard negatives.
- We develop a co-evolving framework that uses target-agent failures to train the failure agent and the resulting hard negatives to improve the target agent.
- We evaluate our framework across three task domains and multiple base models, demonstrating performance gains from incorporating learned hard-negative generation into agent self-improvement.

2 Related Work

Self-evolving agents. Self-evolving agents improve by training on trajectories collected through their own interactions. A common pipeline first fine-tunes the agent on demonstrations (Chen et al., 2023; Zeng et al., 2024) and then updates it using successful or failed rollouts (Yuan et al., 2023; Song et al., 2024; Yuan et al., 2025). Existing methods retain successful rollouts, pair failures with demonstrations, assign feedback to individual steps, or train on trajectories that show how to recover from errors (Yuan et al., 2023; Song et al., 2024; Xiong et al., 2024; Yuan et al., 2025). Online curricula can further adjust the training tasks as the agent improves (Qi et al., 2025). These methods use self-generated trajectories to improve the agent or adapt its training curriculum. However, failures collected from the current agent can become less informative after it learns to avoid earlier error patterns.

Learning from generated preferences. Preference optimization trains a model to prefer one output over another without learning a separate reward model (Rafailov et al., 2023). Recent methods

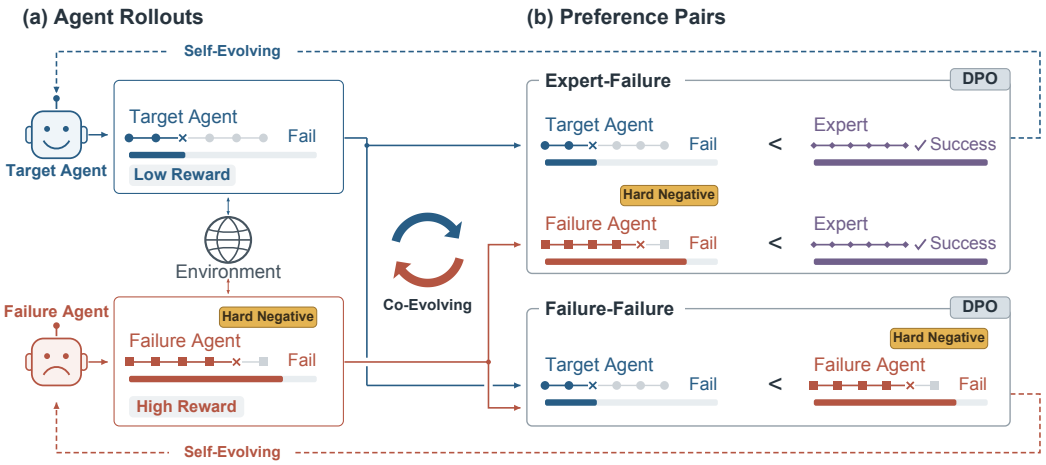

Figure 1: Overview of our co-evolving framework. Both agents generate trajectories for repeated training. The target agent prefers expert demonstrations over failures; the failure agent prefers higher-reward failures over lower-reward ones, regardless of source. The highlighted hard negative is rejected during target agent training but preferred during failure agent training. Colors denote trajectory sources, bars show rewards, and dashed loops indicate repeated updates.

construct preferences from model-generated evaluations, combine feedback at multiple levels of a trajectory, or allow models with different capabilities to learn from one another (Yuan et al., 2024; Gao et al., 2026; Lyu et al., 2025). Negative agents have also been trained on dispreferred responses to supply negatives for alignment; these generators remain fixed after initialization (Lyu et al., 2025).

Hard negatives and sampling. Hard negatives are plausible failures that are more difficult to distinguish from successful behavior than obvious failures. Prior work shows that such negatives can provide stronger contrastive supervision (Robinson et al., 2021) and proposes sampling rejected responses that are difficult for the current policy (Chen et al., 2026). Other work allocates more trials to difficult prompts to collect successful solutions (Tong et al., 2024). Most focus on response-level preference learning or on collecting successful solutions, while hard-negative generation for failed multi-turn interactions remains less explored.

3 Preliminaries

We model agent–environment interaction as a partially observable Markov decision process, following Song et al. (2024). Let $\mathcal{U}$, $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the instruction, state, action, and observation spaces, respectively. The transition model $T(s' \mid s, a)$ describes how an action changes the latent state, and the observation model $Z(o \mid s', a)$ describes the feedback returned to the agent. Instructions, actions, and observations are represented as text.

Given an instruction $u \in \mathcal{U}$, the agent selects each action from its interaction history:

$$h_j = (u, a_1, o_1, \ldots, a_{j-1}, o_{j-1}), \qquad a_j \sim \pi_\theta(\cdot \mid h_j).$$

For the first action, $h_1 = u$, where u includes any initial environment observation. Interaction continues until the episode terminates or the step budget is reached. A trajectory $\tau = (u, a_1, o_1, \ldots, a_n, o_n)$ includes the observation returned after the final action. We write $\pi_\theta(\tau \mid u)$ for the likelihood of the agent's actions conditioned on the recorded observations:

$$\log \pi_\theta(\tau \mid u) = \sum_{j=1}^{n} \log \pi_\theta(a_j \mid h_j). \tag{1}$$

Each action likelihood is evaluated autoregressively over its tokens. Instruction and observation tokens provide context but do not contribute to the training loss.

The environment assigns an episode-level reward $r(u, \tau) \in [0, 1]$ to the trajectory. This reward measures task completion and need not be a sum of step rewards. Each environment specifies a success condition for the task.

Table 1: Benchmark statistics. ScienceWorld includes seen and unseen evaluation splits. Max Turns denotes the interaction limit per episode.

Dataset	Train	Test		Action Space	Max Turns
		Seen	Unseen		
WebShop	1,824	200	–	8	10
ScienceWorld	1,483	194	211	16	100
InterCodeSQL	1,500	200	–	∞	10

4 METHOD

Our framework trains a target agent to solve tasks and a failure agent to generate hard negatives. The target agent learns to prefer expert demonstrations over failed trajectories. The failure agent learns to generate high-reward failures by comparing unsuccessful trajectories from both agents. Both agents start from the same supervised checkpoint and exchange self-generated trajectories over successive training iterations (Figure 1).

4.1 SUPERVISED INITIALIZATION

Supervised fine-tuning on expert demonstrations is a common initialization step before further agent training (Song et al., 2024; Zeng et al., 2024). We use a small set of domain-specific expert demonstrations $\mathcal{D}_{\text{exp}}$ to train an initial policy for the task environment:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(u,\tau_{\text{exp}})\sim\mathcal{D}_{\text{exp}}}\left[\log \pi_\theta(\tau_{\text{exp}} \mid u)\right]. \tag{2}$$

The resulting checkpoint initializes both the target agent π_θ and the failure agent π_ϕ, giving them the same starting point before they learn from different preference data. Demonstration sources are described in Appendix A.

4.2 LEARNING TO GENERATE HARD NEGATIVES

We train the failure agent to generate high-reward failures that serve as hard negatives for training the target agent. Environment reward measures partial task completion, which we use as a proxy for negative difficulty. This proxy does not directly measure how difficult a failure is for the target agent to distinguish from success.

We obtain the training signal by comparing failures from the two agents on the same instruction. Let $\mathcal{F}_\theta(u)$ and $\mathcal{F}_\phi(u)$ denote their respective sets of failed trajectories for instruction u. For each pair $\tau_\theta \in \mathcal{F}_\theta(u)$ and $\tau_\phi \in \mathcal{F}_\phi(u)$, we prefer the trajectory with higher reward:

$$(\tau^+, \tau^-) = \begin{cases} (\tau_\theta, \tau_\phi), & r(u,\tau_\theta) > r(u,\tau_\phi), \\ (\tau_\phi, \tau_\theta), & r(u,\tau_\phi) > r(u,\tau_\theta). \end{cases} \tag{3}$$

We discard ties and collect preference triplets (u, τ^+, τ^-) in $\mathcal{D}_{\text{fail}}$. Both τ^+ and τ^- are failed trajectories, with $r(u,\tau^+) > r(u,\tau^-)$. The preferred trajectory can come from either agent.

We optimize the failure agent on $\mathcal{D}_{\text{fail}}$ using Direct Preference Optimization (DPO) (Rafailov et al., 2023). For a policy π and a reference policy π_{ref}, the loss for a preference pair is

$$\ell_{\text{DPO}}(\pi; u, \tau^+, \tau^-) = -\log \sigma\left[\beta\left(\log \frac{\pi(\tau^+ \mid u)}{\pi_{\text{ref}}(\tau^+ \mid u)} - \log \frac{\pi(\tau^- \mid u)}{\pi_{\text{ref}}(\tau^- \mid u)}\right)\right]. \tag{4}$$

Here, σ is the logistic sigmoid and $\beta > 0$ is a scaling parameter. Each agent uses its own reference policy, held fixed during each optimization phase. We train the failure agent by minimizing

$$\mathcal{L}_{\text{fail}}(\phi) = \mathbb{E}_{(u,\tau^+,\tau^-)\sim\mathcal{D}_{\text{fail}}}\left[\ell_{\text{DPO}}(\pi_\phi; u, \tau^+, \tau^-)\right]. \tag{5}$$

The failure agent is trained to prefer higher-reward failures using only failure–failure pairs. After the shared supervised initialization, its updates use no expert demonstrations or auxiliary SFT loss. The failure agent may still generate successful trajectories, which are excluded from its training data.

4.3 Learning from Hard Negatives

We augment the target agent's own failures with hard negatives generated by the failure agent. For each instruction, we pair each collected failure with an expert demonstration, using the demonstration as the preferred trajectory and the failure as the rejected trajectory. This gives the target preference dataset

$$\mathcal{D}_{\mathrm{tgt}} = \left\{(u, \tau_{\mathrm{exp}}, \tau^-) : (u, \tau_{\mathrm{exp}}) \in \mathcal{D}_{\mathrm{exp}},\ \tau^- \in \mathcal{F}_\theta(u) \cup \mathcal{F}_\phi(u)\right\}. \tag{6}$$

We apply the DPO loss in Equation 4 to these pairs. DPO learns relative preferences, but its updates can become dominated by suppressing rejected responses without sufficiently reinforcing preferred responses (Yan et al., 2025). To stabilize target training, we add an auxiliary SFT loss on the expert demonstrations. This term directly increases the likelihood of expert actions, complementing the relative preference signal from DPO.

We normalize pair weights within each instruction so that instructions with more failures do not dominate training. Let $\mathbb{E}_w$ denote the weighted average over $\mathcal{D}_{\mathrm{tgt}}$, with equal total weight for each instruction (Appendix B.2). The combined objective is

$$\mathcal{L}_{\mathrm{tgt}}(\theta) = \mathbb{E}_w\left[\ell_{\mathrm{DPO}}(\pi_\theta; u, \tau_{\mathrm{exp}}, \tau^-)\right] + \lambda_{\mathrm{SFT}}\,\mathbb{E}_w\left[-\log \pi_\theta(\tau_{\mathrm{exp}} \mid u)\right]. \tag{7}$$

Here, λ_{SFT} controls the weight of the auxiliary SFT loss. By incorporating failures from a separately trained failure agent, target training extends beyond the target agent's own mistakes, which may help it generalize beyond the failure patterns encountered in its own rollouts.

4.4 Co-Evolution through Trajectory Exchange

We train the two agents iteratively, starting from the same supervised checkpoint. In each iteration, both agents collect trajectories through environment interaction. We construct $\mathcal{D}_{\mathrm{fail}}$ and $\mathcal{D}_{\mathrm{tgt}}$ from the collected failures and expert demonstrations, as described in Sections 4.2 and 4.3. Each agent is then updated with its respective objective in Equations 5 and 7. We repeat this procedure using trajectories generated by the updated agents.

The failure agent learns from the target agent's current failures, so its training data change as the target agent is updated. In turn, the updated failure agent generates new negatives for subsequent target training. This feedback loop allows hard-negative generation to adapt as the target agent learns. Both agents therefore learn from their own interactions with the environment and from trajectories generated by the other agent.

The failure agent supports the target agent during training by generating hard negatives for preference learning. At inference time, we use only the target agent to interact with the environment and complete tasks.

5 Experiments

We evaluate whether learning to generate hard negatives improves agent self-improvement across three task domains and multiple base models. Performance comparisons assess the full framework, trajectory analyses examine the generated training data, and ablations test the roles of the failure agent, continued co-evolution, and auxiliary expert supervision.

5.1 Experimental Setup

We consider a setting with a small set of expert demonstrations for each environment.

Environments. We evaluate on three interactive environments. WebShop requires agents to search for products and purchase an item that satisfies the user's requirements (Yao et al., 2022). ScienceWorld requires agents to carry out scientific procedures that combine navigation, object manipulation, and reasoning across multiple steps (Wang et al., 2022). InterCodeSQL requires agents to answer database questions by generating SQL queries and refining them using execution feedback (Yang et al., 2023). Trajectories span up to 10 turns in WebShop and InterCodeSQL and up to 100 turns in ScienceWorld. All three environments return rewards in $[0, 1]$ and provide partial credit for task

Table 2: Main results on WebShop, ScienceWorld, and InterCodeSQL across three base models. Results are average task rewards on a 0–100 scale.

Model	Method	WebShop	ScienceWorld		InterCodeSQL	Avg.
			Seen	Unseen		
Qwen3-4B	SFT	66.00	83.30	67.90	58.10	68.83
	ETO	72.90	80.70	83.10	60.70	74.35
	Ours	**74.10**	**87.60**	**88.30**	**63.00**	**78.25**
Qwen3.5-9B	SFT	55.15	79.24	73.80	68.59	69.20
	Agent-R	55.24	69.68	56.66	65.31	61.72
	RFT	56.41	80.00	74.57	69.57	70.14
	ETO	68.06	80.47	75.40	68.94	73.22
	Ours	**71.30**	**88.60**	**84.72**	**70.63**	**78.81**
Gemma-4-12B-it	SFT	71.10	70.71	64.96	66.77	68.39
	Agent-R	69.70	70.21	65.19	67.32	68.10
	RFT	72.22	77.25	73.29	70.82	73.40
	ETO	71.67	77.11	69.59	70.36	72.18
	Ours	**73.49**	**87.55**	**84.42**	**73.38**	**79.71**

progress. We report average rewards on a 0–100 scale. ScienceWorld includes separate seen and unseen evaluation splits. Table 1 summarizes dataset sizes and interaction budgets, and Appendix A describes the reward rules and demonstration sources.

Models. We evaluate Qwen3-4B-Instruct-2507 (Yang et al., 2025), Qwen3.5-9B (Qwen Team, 2026), and Gemma-4-12B-it (Google DeepMind, 2026). To assess whether the gains extend to earlier model generations, we also evaluate Llama-2-7B-Chat and Llama-2-13B-Chat (Touvron et al., 2023), with results reported in Appendix C. The target and failure agents use the same model architecture and start from the same supervised checkpoint.

Baselines. SFT is the supervised starting policy. RFT trains on successful generated trajectories (Yuan et al., 2023), ETO uses expert–failure preferences (Song et al., 2024), and Agent-R uses iterative self-training for error recovery (Yuan et al., 2025). Self-Rewarding Language Models (SRLM) construct preferences through self-evaluation of generated candidates (Yuan et al., 2024). To compare learned hard-negative generation with additional sampling, Best-of-N samples four trajectories from the target agent for each instruction and selects the one with the highest environment reward for preference construction. PPO (Schulman et al., 2017) and prompt-only GPT baselines appear in the additional model comparisons in Appendix C.

Implementation details. We initialize both agents with SFT on expert demonstrations and use AdamW (Loshchilov & Hutter, 2019) with three epochs per optimization phase. The default target objective uses $\lambda_{\text{SFT}} = 0.1$; Table 5 varies this weight. Multiple-candidate comparisons use $N = 4$. Rollout counts are defined per instruction per generating agent and do not imply matched total training costs. Evaluation uses the target agent alone. Avg. is the arithmetic mean of the four evaluation columns. Further optimization settings and computational resources are given in Appendix B.

5.2 Main Results

Table 2 reports task performance across three environments and three base models. ETO trains the target agent on expert–failure pairs using its own failed trajectories. The comparison with ETO evaluates the benefit of learning to generate hard negatives with a separate failure agent.

Our framework achieves the highest task reward across the evaluated models and environments, with an average improvement of 5.7 over ETO. Performance improves across all three environments, especially on ScienceWorld, the most challenging long-horizon environment in our evaluation. Its tasks combine scientific reasoning with multiple dependent actions over trajectories of up to 100 turns. These results suggest that a trained failure agent provides useful supervision beyond the target agent's own failures, supporting agent self-improvement across models and tasks.

Table 3: Comparison with methods that use multiple rollouts during training. SRLM and Best-of-N use $N = 4$; ETO and Ours use one rollout per instruction per generating agent.

Model	Method	WebShop	ScienceWorld		InterCodeSQL	Avg.
			Seen	Unseen		
Qwen3.5-9B	ETO	68.06	80.47	75.40	68.94	73.22
	SRLM	54.09	78.11	74.09	66.34	68.16
	Best-of-N	71.14	82.87	75.94	65.02	73.74
	Ours	**71.30**	**88.60**	**84.72**	**70.63**	**78.81**
Gemma-4-12B-it	ETO	71.67	77.11	69.59	70.36	72.18
	SRLM	72.27	72.93	68.52	71.75	71.37
	Best-of-N	71.44	80.72	72.64	71.05	73.96
	Ours	**73.49**	**87.55**	**84.42**	**73.38**	**79.71**

5.3 Comparison with Additional Sampling

To evaluate whether additional sampling can match the benefits of learned hard-negative generation, we compare our framework with SRLM and Best-of-N under the ETO training setup. Both methods sample four target trajectories per instruction. SRLM assigns rewards through self-evaluation and uses them to construct preference pairs without environment rewards. Best-of-N instead uses environment rewards to select the highest-reward trajectory for preference construction.

Our framework achieves the highest task rewards across both models and all environments (Table 3). SRLM constructs pairs using self-assigned rewards and generally underperforms methods using environment rewards. Best-of-N selects the highest-reward rollouts, but those matching the expert reward yield no valid expert–failure pairs and can reduce opportunities to learn from failures. These results support training a failure agent to generate informative negatives for self-improvement.

5.4 Qualitative Results

We analyze matched ScienceWorld examples to compare target performance and the failures used for training. ETO learns from its own failures and is among the strongest baselines in Table 2. Both comparisons below use Gemma-4-12B-it.

Target Performance: Pollination (ScienceWorld)

Instruction: Grow an orange by growing several plants and allowing cross-pollination. Focus on the grown orange to finish the task.

ETO target Reward 0.446 FAILURE

Plants three orange seeds and waits until the trees have flowers, but **leaves the bee hive closed**. It continues waiting and attempts to focus on an orange before any fruit appears, leaving the pollination step incomplete.

Ours target Reward 1.000 SUCCESS

Plants the seeds, waits for flowers, and **opens the bee hive**. Later observations show a bee and oranges on the trees; after resolving an ambiguous object selection, it **focuses on a grown orange and completes the task**.

Key difference: Both agents grow flowering trees, but only Ours enables pollination and follows the resulting fruit through to task completion.

Training Negatives: Measurement and Decision (ScienceWorld)

Instruction: Measure the melting point of gallium. Focus on the orange box if it is above 200°C, or the yellow box if it is below 200°C.

ETO negative Reward 0.403 FAILURE

Retrieves and focuses on the thermometer, then attempts to focus on the gallium-containing jug. The environment **selects the jug itself**, and the rollout **ends before the agent measures the material** or selects an answer box.

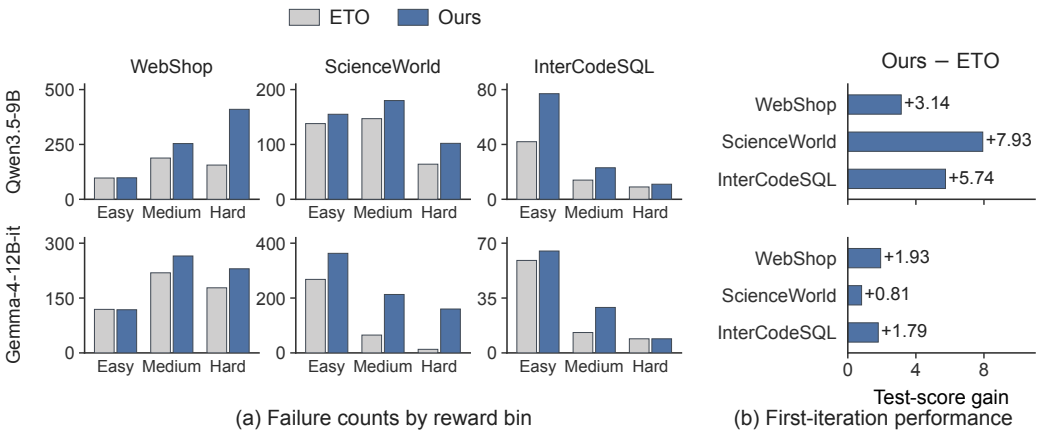

Figure 2: Failure rewards and early target performance. (a) Failure counts by reward bin for Qwen3.5-9B (top) and Gemma-4-12B-it (bottom). Each panel uses its own count scale. (b) Target performance gains over ETO at iteration 1. Reward bins are used only for analysis.

Training Negatives: Measurement and Decision (ScienceWorld) (continued)

Ours hard negative Reward 0.747 **FAILURE**

Focuses on gallium, opens the freezer, cools the sample, and obtains a 7°C reading. It nevertheless **selects the orange box for the above-200°C condition**; the measurement also **does not establish the melting point** requested by the instruction.

Key difference: The hard negative completes more of the procedure but still reaches an unsupported conclusion. The error concerns what the measurement establishes and how it should guide the final decision.

The target comparison highlights a missing dependency in a multi-step task: flowering must be followed by pollination to produce fruit. The training comparison shows a different distinction: completing several experimental steps does not ensure a valid conclusion. Such failures offer a concrete contrast with successful demonstrations beyond early procedural errors. Appendix D provides three comparisons per environment, including the corresponding actions and observations.

5.5 Failure Reward Analysis

We examine whether the failure agent provides more high-reward negatives for target training than ETO. Figure 2(a) groups failures into Easy, Medium, and Hard bins, corresponding to the lower, middle, and upper thirds of the reward range, respectively. These bins are used only for analysis.

Our framework produces more Hard failures in most model–environment settings, providing a larger pool of high-reward negatives for target training. Their proportion increases on WebShop and ScienceWorld but decreases on InterCodeSQL, indicating that greater availability does not imply a uniform shift in the reward distribution. Panel (b) shows target performance gains over ETO across all evaluated settings. This analysis characterizes the training data but does not isolate their effect on performance.

5.6 Component Ablations

Failure agent. To assess the contribution of the failure agent, we replace it with a positive agent trained with the same objective as the target agent. We keep the total parameter count unchanged. This separate positive agent provides additional trajectories to augment the target agent's training data. As shown in Table 4, training with a failure agent yields higher target agent reward than training with a positive agent across the reported settings, with the largest gains on ScienceWorld's multi-step scientific tasks. These results support training the additional agent specifically to generate hard negatives for the target agent.

Co-evolution. To evaluate the effect of co-evolution, we freeze the failure agent after training it in iteration 1. The failure agent has thus learned from the target agent's initial failures, but does not learn from the target agent's subsequent failures. The target agent continues learning from new negatives generated by this frozen failure agent. As shown in Table 4, continuing to update the failure agent yields higher target agent reward than freezing it across all reported settings, with larger gains on ScienceWorld. These results suggest that learning from the target agent's recent failures helps the failure agent generate negatives that are more informative for the target agent at its current stage of learning.

Table 4: Ablations of the failure agent and co-evolution. w/o Failure Agent replaces the failure agent with a positive agent trained with the target objective; w/o Co-Evolution freezes the failure agent after iteration 1.

Model	Method	WebShop	ScienceWorld		InterCodeSQL	Avg.
			Seen	Unseen		
Qwen3.5-9B	SFT	55.15	79.24	73.80	68.59	69.20
	ETO	68.06	80.47	75.40	68.94	73.22
	Ours	**71.30**	**88.60**	**84.72**	**70.63**	**78.81**
	w/o Co-Evolution	71.10	87.05	79.52	70.27	76.99
	w/o Failure Agent	70.93	68.62	63.03	68.71	67.82
Gemma-4-12B-it	SFT	71.10	70.71	64.96	66.77	68.39
	ETO	71.67	77.11	69.59	70.36	72.18
	Ours	**73.49**	**87.55**	**84.42**	**73.38**	**79.71**
	w/o Co-Evolution	73.21	81.29	76.43	71.84	75.69
	w/o Failure Agent	72.99	77.33	72.52	71.38	73.56

5.7 Sensitivity to the SFT Weight

The auxiliary SFT loss reinforces expert actions to stabilize DPO training. We use $\lambda_{\mathrm{SFT}} = 0.1$ across all models and environments in the main experiments and vary this weight on ScienceWorld to assess sensitivity (Table 5). Both models score highest at the shared weight; increasing it does not improve performance. This supports using SFT as a moderately weighted auxiliary objective. Appendix C.3 reports the Llama-2 results on WebShop.

Table 5: SFT weight ablation.

Model	Split	λ_{SFT} 0.01	0.1	0.5
Qwen3.5	Seen	85.17	**88.60**	87.90
	Unseen	80.56	**84.72**	76.41
Gemma-4	Seen	84.84	**87.55**	87.05
	Unseen	83.65	**84.42**	83.89

6 Conclusion

We introduced a co-evolving framework that learns to generate hard negatives for agent self-improvement. The failure agent learns from both agents' failed trajectories to generate higher-reward failures, which the target agent uses for preference training. As the target agent improves, its new failures guide further learning by the failure agent, allowing negative generation to adapt throughout training. Experiments on WebShop, ScienceWorld, and InterCodeSQL show improvements over baselines, especially on ScienceWorld's challenging long-horizon tasks. The failure agent yields higher target reward than an additional positive agent used to augment the training data. These findings highlight the value of learning to generate informative negatives for agent self-improvement.

Limitations. Training the failure agent increases rollout and optimization costs, although inference uses only the target agent. The failure agent relies on reward differences, which provide no preference signal for tied failures and do not directly measure negative difficulty. Future work could explore self-verification to identify informative failures without relying on environment rewards.

AI USE STATEMENT

We used LLMs or other AI tools to aid or polish writing, for research ideation or execution, to draft sections of the paper, and for the implementation of experimental code. The authors guided and reviewed all AI-assisted work and take full responsibility for the final manuscript and associated artifacts.

REPRODUCIBILITY STATEMENT

Section 4 describes trajectory construction, the construction of preference pairs, and the learning objectives for both agents. Section 5.1 specifies the benchmarks, evaluated models, baselines, and evaluation metrics. Appendix A documents the demonstration sources and reward rules, Appendix B summarizes the optimization settings and computational resources, and Appendix B.2 details the sample weights and supervision from expert demonstrations used to train the target agent. The accompanying implementation details distinguish the agents' objectives, the trajectory sources, and the supervision used for each update.

REFERENCES

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FiReAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9354–9366. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.557. URL https://aclanthology.org/2024.findings-acl.557/.

Zhuotong Chen, Fang Liu, Xuan Zhu, Haozhu Wang, Jiayu Li, Yanjun Qi, and Mohammad Ghavamzadeh. Preference optimization via contrastive divergence: Your policy is secretly an NLL estimator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 37286–37294, 2026. doi: 10.1609/aaai.v40i44.41060. URL https://ojs.aaai.org/index.php/AAAI/article/view/41060.

Heyang Gao, Zexu Sun, Erxue Min, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Xu Chen. Solving the granularity mismatch: Hierarchical preference learning for long-horizon LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=s8usvGHYlk.

Google DeepMind. Gemma-4-12B-it. https://huggingface.co/google/gemma-4-12B-it, 2026. Model card; accessed 2026-08-20.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yougang Lyu, Lingyong Yan, Zihan Wang, Dawei Yin, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. MACPO: Weak-to-strong alignment via multi-agent contrastive preference optimization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=x1Okv4kbVR.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c66e1fcc9691aae706250638f36f681b-Paper-Conference.pdf.

Qwen Team. Qwen3.5-9B. https://huggingface.co/Qwen/Qwen3.5-9B, 2026. Model card; accessed 2026-08-20.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=CR1XOQ0UTh-.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-Math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems*, volume 37, pp. 7821–7846, 2024. doi: 10.52202/079017-0251. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0ef1afa0daa888d695dcd5e9513bafa3-Abstract-Conference.html.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1556–1572, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.93. URL https://aclanthology.org/2024.emnlp-main.93/.

Yuzi Yan, Yibo Miao, Jialian Li, Yipin Zhang, Jian Xie, Zhijie Deng, and Dong Yan. 3D-Properties: Identifying challenges in DPO and charting a path forward. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c080f9fb3658511bc4f063c5794ef706-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. https://arxiv.org/abs/2505.09388, 2025. Accessed: 2025-09-24.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. InterCode: Standardizing and benchmarking interactive coding with execution feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1035. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract.html. Datasets and Benchmarks Track.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

A Environments and Rewards

We describe the task objectives, reward signals, and demonstration sources for the three environments. All experiments use episode-level rewards in $[0, 1]$. We use a maximum interaction budget of 10 turns per episode for WebShop and InterCodeSQL and 100 turns for ScienceWorld. Dataset sizes and interaction budgets are summarized in Table 1.

A.1 WebShop

WebShop requires an agent to select a product and options that satisfy a shopping instruction (Yao et al., 2022). Its terminal reward combines matches with the requested attributes and options, satisfaction of the price constraint, and a match with the requested product type. Partial matches receive partial credit. We use the demonstration trajectories provided by Song et al. (2024).

A.2 ScienceWorld

ScienceWorld requires navigation, object manipulation, and scientific procedures (Wang et al., 2022). We normalize its task progress score to $[0, 1]$. The score reflects completion of intermediate goals, so unsuccessful trajectories can receive partial credit. Following the evaluation setup used in our experiments, we exclude the two longest task types and subsample test instances with more than ten variations. Seen scenarios and unseen variations are evaluated separately. Demonstrations are obtained from the official data and GPT-4-generated trajectories following Song et al. (2024).

A.3 InterCodeSQL

InterCodeSQL requires an agent to answer a database question through interactive SQL execution (Yang et al., 2023). The reward compares the final execution output with the reference output using an intersection-over-union score that accounts for matching records, duplicates, and record order. Invalid execution outputs receive zero reward, partial matches receive intermediate rewards, and an exact correctly ordered match receives one. We use GPT-4 demonstrations collected by Xiong et al. (2024), retaining those with reward 1.0, and add ReAct-style thought annotations following that setup (Yao et al., 2023).

B Training Details

B.1 Optimization

Both agents are initialized from the same supervised checkpoint. The recorded SFT configuration uses AdamW for three epochs with batch size 48, learning rate 10^{-5}, a 3% linear warmup, and cosine decay. Preference optimization uses batch size 32 and three epochs per phase, with learning rates between 5×10^{-7} and 10^{-6} and DPO scale β between 0.1 and 0.5. Experiments use eight NVIDIA H100 GPUs with 80 GB memory each. The main target objective uses $\lambda_{\mathrm{SFT}} = 0.1$.

B.2 Preference Data and Sample Weighting

The target agent is trained on expert–failure pairs: the expert demonstration is preferred, and the paired failure comes from either agent. The auxiliary SFT term uses the expert demonstration from the same pair. The failure agent uses only failure–failure preferences and has no auxiliary SFT term after initialization.

To give each instruction equal total weight, let $m_u > 0$ denote the number of retained target preference pairs for instruction u. Starting from uniform pair weights, we assign each pair weight $1/m_u$. If w_i is the resulting weight of pair i, the empirical average in Equation 7 is

$$\mathbb{E}_w[f] = \frac{\sum_i w_i f_i}{\sum_i w_i}. \tag{8}$$

This normalization prevents instructions with more collected failures from contributing more total weight. It is applied to both the target DPO term and its auxiliary SFT term.

C Additional Experiments

C.1 Additional Model Families

We report additional comparisons using the earlier Llama-2-7B-Chat and Llama-2-13B-Chat models in Tables 6 and 7. Our framework exceeds ETO in every reported task column for both models. Each table reports a separate experimental configuration; the results should be compared within the corresponding setting. GPT results in Table 6 use in-context learning rather than task-specific fine-tuning.

Table 6: Additional results for Llama-2-7B-Chat. GPT baselines use in-context learning. Avg. is the mean over the four evaluation settings.

Adaptation	Models	WebShop	ScienceWorld		InterCodeSQL	Avg.
			Seen	Unseen		
In-context	GPT-4	63.2	42.9	38.1	38.5	45.7
	GPT-3.5-Turbo	62.4	7.9	10.5	37.8	29.7
Fine-tuning	Llama-2-7B-Chat + SFT	58.7	53.4	49.9	26.5	47.1
	Llama-2-7B-Chat + PPO	64.2	59.4	51.7	32.4	51.9
	Llama-2-7B-Chat + RFT	61.3	71.6	54.3	25.6	53.2
	Llama-2-7B-Chat + ETO	63.0	73.1	65.3	32.3	58.4
	Llama-2-7B-Chat + Ours	**70.3**	**82.4**	**69.2**	**43.4**	**66.3**

Table 7: Additional results for Llama-2-13B-Chat on WebShop and ScienceWorld. Avg. is the mean over the two environments.

Models	WebShop	ScienceWorld	Avg.
Llama-2-13B-chat + SFT	61.8	64.5	63.2
Llama-2-13B-chat + AgentLM (Zeng et al., 2024)	39.5	10.7	25.1
Llama-2-13B-chat + Agent-FLAN (Chen et al., 2024)	40.4	28.6	34.5
Llama-2-13B-chat + ETO	66.0	72.6	69.3
Llama-2-13B-chat + Ours	**69.2**	**74.5**	**71.9**

C.2 Additional Sampling for RFT

We examine whether allocating more samples to difficult instructions improves RFT, which learns from successful trajectories. A DART-style procedure uses five initial rollouts to estimate instruction difficulty and allocates a total of ten candidates per instruction. Successful candidates augment the supervised training data. In this separate WebShop experiment, the score increases from 60.89 to 61.90, a gain of 1.01 points. The result supports additional sampling for collecting successful training trajectories in this setting. It addresses a different question from Best-of-N, which selects trajectories for preference learning, and uses a separate configuration from the RFT results in the main table.

C.3 Sensitivity to the SFT Weight

Table 8 reports sensitivity to the SFT weight for Llama-2-7B on WebShop. The highest score occurs at $\lambda_{\mathrm{SFT}} = 0.01$; increasing the weight does not improve performance over the tested range.

Table 8: Effect of the SFT weight on Llama-2-7B performance in WebShop.

λ_{SFT}	WebShop
0.01	67.98
0.1	66.70
0.5	66.50

D QUALITATIVE RESULTS FOR QWEN3.5 AND GEMMA-4

We present three matched comparisons per environment: two target evaluations and one comparison of training failures. Target evaluations compare ETO with our target agent; training comparisons contrast failures from ETO's target agent with those from our failure agent. Each case reports the model, task ID, recorded reward, and outcome. Rewards are on the original $[0, 1]$ scale. We summarize the behavior and show selected actions and environment responses. ScienceWorld target examples use the unseen split. Evaluation trajectories are from `data/qualitative_trajs`; training failures are from `data/train_failures`.

D.1 SCIENCEWORLD

Completing the Pollination Step

Setting: Gemma-4-12B-it; task `15_93`. Target evaluation.

Instruction: Grow an orange by growing several plants and allowing cross-pollination. Focus on the grown orange to finish the task.

ETO target Reward 0.446 **FAILURE**

Plants three orange seeds and waits until the trees have flowers, but **leaves the bee hive closed**. It continues waiting and attempts to focus on an orange before any fruit appears, leaving the pollination step incomplete.

Environment: "The bee hive door is closed." Later attempts to `focus on orange in reproducing orange tree` receive "No known action matches that input."

Ours target Reward 1.000 **SUCCESS**

Plants the seeds, waits for flowers, and **opens the bee hive**. Later observations show a bee and oranges on the trees; after resolving an ambiguous object selection, it **focuses on a grown orange and completes the task**.

Action: `open bee hive`. Later observations list an adult bee and oranges; `focus on orange` followed by object selection yields "You focus on the orange."

Key difference: Both agents grow flowering trees, but only Ours enables pollination and follows the resulting fruit through to task completion.

Using Feedback to Complete a Circuit

Setting: Qwen3.5-9B; task `10_451`. Target evaluation.

Instruction: Test whether unknown substance U conducts electricity. Place it in the green box if conductive, or the blue box otherwise.

ETO target Reward 0.643 **FAILURE**

Builds part of a battery-and-bulb circuit, then repeatedly tries to connect wires to **an anode or cathode on the substance**. The environment rejects these actions; repeated waiting does not repair the connection, and the material is not classified.

Action: `connect blue wire terminal 2 to cathode in unknown substance U`. Response: "No known action matches that input."

Ours target Reward 1.000 **SUCCESS**

Connects the substance through its **numbered terminals** and completes the circuit. After waiting, it observes that **the red bulb is on** and moves the substance to the green box, linking the observed electrical response to the required decision.

Connections use `unknown substance U terminal 1` and `terminal 2`. The room description changes to "a red light bulb, which is on"; the environment then confirms placement in the green box.

Key difference: Ours completes the experimental setup and uses its observed outcome to classify the material, whereas ETO repeats an invalid connection.

A Measurement without a Valid Conclusion

Setting: Gemma-4-12B-it; task `4_22`. Training failures, iteration 2.

Instruction: Measure the melting point of gallium. Focus on the orange box if it is above 200°C, or the yellow box if it is below 200°C.

ETO negative Reward 0.403 FAILURE

Retrieves and focuses on the thermometer, then attempts to focus on the gallium-containing jug. The environment **selects the jug itself**, and the rollout **ends before the agent measures the material** or selects an answer box.

Action: `focus on jug containing gallium`. Response: "You focus on the jug." No thermometer reading follows.

Ours hard negative Reward 0.747 FAILURE

Focuses on gallium, opens the freezer, cools the sample, and obtains a 7°C reading. It nevertheless **selects the orange box for the above-200°C condition**; the measurement also **does not establish the melting point** requested by the instruction.

Action: `use thermometer in inventory on gallium`. Response: "the thermometer measures a temperature of 7 degrees celsius". The next action is `focus on orange box`.

Key difference: The hard negative completes more of the procedure but still reaches an unsupported conclusion. The error concerns what the measurement establishes and how it should guide the final decision.

D.2 WEBSHOP

Preserving the Requested Size

Setting: Gemma-4-12B-it; task `11151`. Target evaluation.

Instruction: Buy men's size-14 sneakers with a synthetic sole for less than $140.

ETO target Reward 0.667 FAILURE

Opens a Madden sneaker listing whose available sizes stop at 13. It **selects size 13** and buys the product, completing the purchase flow but dropping the explicit size requirement from the instruction.

Product: `B08L9SFY3C`. Actions include `click[13]` and `click[Buy Now]`.

Ours target Reward 1.000 SUCCESS

Selects an ALDO sneaker listing that **offers size 14, chooses that size**, and completes the purchase. The recorded actions preserve the requested size through product selection and checkout rather than substituting an available smaller option.

Product: `B08SPLLMN3`. Actions include `click[14]` and `click[Buy Now]`.

Key difference: The distinction is the selected product option: completing a purchase does not suffice when the requested size is unavailable.

Checking the Budget before Purchase

Setting: Qwen3.5-9B; task `1414`. Target evaluation.

Instruction: Buy green, modern velvet dining chairs for less than $210.

ETO target Reward 0.667 FAILURE

Finds green velvet dining chairs, selects the green option, and **buys a set listed at $260.99**. The product matches the requested style and color, but its displayed price exceeds the budget before checkout.

Product: `B07YTMKL9T`; displayed price: $260.99. Actions: `click[green]`, then `click[Buy Now]`.

Checking the Budget before Purchase (continued)

Ours target Reward 1.000 SUCCESS

Chooses a different velvet dining-chair listing **priced at $173.99**, selects emerald green, and completes the purchase. This keeps the requested appearance while satisfying the price constraint, rather than accepting the first visually relevant product.

Product: `B07R9KFCKY`; displayed price: $173.99. Actions: `click[emerald green]`, then `click[Buy Now]`.

Key difference: Both agents choose relevant chairs; the successful trajectory also respects the budget at the purchase decision.

Relevant Shoes with a Missed Budget Constraint

Setting: Gemma-4-12B-it; task `7092`. Training failures, iteration 1.

Instruction: Buy breathable, slip-resistant men's shoes with rubber soles, size 8, for less than $80.

ETO negative Reward 0.075 FAILURE

Selects a hiking sneaker listed at $42.88, chooses size 8 and black, and completes the purchase. Although the recorded size and price satisfy those two requirements, the overall product match receives only a small partial reward.

Product: `B07S7QXNCW`; price: $42.88. Actions include `click[8]`, `click[black]`, and `click[Buy Now]`.

Ours hard negative Reward 0.750 FAILURE

Chooses shoes described as **breathable and non-slip, selects size 8**, and buys them. The listing is **priced at $100**, so the trajectory preserves several relevant choices but **fails to enforce the explicit budget constraint** at checkout.

Product: `B09CYSLPZL`; title includes "Breathable Non-Slip"; price: $100. Actions include `click[8]` and `click[Buy Now]`.

Key difference: The higher-reward failure is a plausible purchase with a specific unmet constraint. Comparing it with a successful demonstration can expose the missed price check.

D.3 INTERCODESQL

Returning All Requested Fields

Setting: Gemma-4-12B-it; task `82`. Target evaluation.

Instruction: Find the language used by the fewest TV channels, and return both the language and the number of channels.

ETO target Reward 0.000 FAILURE

Groups channels by language and orders the groups by their counts, but **selects only the language column**. The query executes and returns English; the agent submits this incomplete output without the requested channel count.

Query fragment: `SELECT Language FROM TV_Channel GROUP BY Language ORDER BY count(*) LIMIT 1`. Output: `[('English',)]`.

Ours target Reward 1.000 SUCCESS

Initially obtains the same language-only output, then **revises the projection to include the count**. It submits the resulting language–count pair, correcting the missing output field while retaining the grouping and ordering needed by the question.

Query fragment: `SELECT Language, COUNT(*) FROM TV_Channel`. The final output is `[('English', 3)]`.

Key difference: Successful execution alone does not establish task success: the returned fields must also match the question.

Using the Stored Categorical Value

Setting: Gemma-4-12B-it; task `114`. Target evaluation.

Instruction: How many dog pets are raised by female students?

ETO target Reward 0.000 FAILURE

Repairs initial query errors and joins the student, ownership, and pet tables. Its **final filter uses `Sex = 'female'`**; this query executes but returns zero, and the agent submits that result without correcting the stored category value.

Final predicates: `student.Sex = 'female'` and `pets.PetType = 'dog'`. Output: `[(0,)]`.

Ours target Reward 1.000 SUCCESS

Inspects the table schemas and joins the same entities using their identifiers. It **filters students with `Sex = 'F'`** and pets with `PetType = 'dog'`, obtaining and submitting a count of two.

The schema gives `Sex` as `varchar(1)`. The final query uses `Sex = 'F'` and returns `[(2,)]`.

Key difference: A valid join can still yield an incorrect answer when the filter uses a natural-language label instead of the value stored in the database.

A Plausible Exclusion Query with the Wrong Identifier

Setting: Gemma-4-12B-it; task `2351`. Training failures, iteration 2.

Instruction: Return the titles and directors of films that were never presented in China.

ETO negative Reward 0.000 FAILURE

Inspects the schemas after two invalid queries, then filters joined records for China while **requiring the film title to be null**. This does not express the requested exclusion, and the submitted query returns an empty result.

Final predicates include `T3.country = 'China'AND T1.title IS NULL`. Output: `[]`.

Ours hard negative Reward 0.770 FAILURE

Uses a plausible `NOT IN` query and returns film titles and directors, earning substantial partial credit. However, it **compares the numeric market identifier with `'CH'`** instead of resolving China through the market table, so the exclusion condition is incorrect.

Exclusion subquery: `SELECT film_id FROM film_market_estimation WHERE market_id = 'CH'`. The schema identifies `Market_ID` as an integer; the submitted output contains 13 title–director pairs.

Key difference: The hard negative produces a nonempty, partially correct answer with the expected columns, but a mistaken identifier invalidates the condition that determines which rows belong in the result.

E Qualitative Results for Llama-2

This section presents qualitative examples from the Llama-2 experiments, complementing the Qwen3.5 and Gemma-4 comparisons in Appendix D. We preserve the reported rewards and step counts, but do not infer exact failure causes or answer correctness from partial reward alone.

E.1 WebShop

HDMI Cables under $50

Instruction: I'm looking for ten high-speed, gold-plated HDMI cables, with price lower than $50.00.

ETO: Selects a single ProHT HDMI cable priced at $100 and proceeds to purchase. The selected quantity and price do not match the instruction.
Reward: 0.50 *Steps:* 4 *Outcome:* Failure

HDMI Cables under $50 (continued)

Ours: Searches for multi-pack HDMI cables, inspects a QualGear candidate, and selects a variant. The summary reports checks of cable properties and price, but the exact pack quantity remains unresolved.
Reward: 0.75 *Steps:* 5 *Outcome:* Failure

Interpretation: The higher-reward failure includes more directed product inspection.

Solid Wood Storage Bench in Grey

Instruction: I want a solid wood bench with storage space for my living room, grey in color, and under $210.00.

ETO: Selects a grey accent bench after a short search and proceeds to purchase.
Reward: 0.50 *Steps:* 3 *Outcome:* Failure

Ours: Inspects several candidates and selects a grey storage bench, with reasoning about its material, storage features, and price.
Reward: 0.75 *Steps:* 5 *Outcome:* Failure

Interpretation: The traces differ in product inspection and selection. The reported reward does not verify all product attributes or establish that price is the sole unmet requirement.

Machine-Washable Curtains (52"×90")

Instruction: I need a machine-washable curtain for the living room, sized 52" wide by 90" long, priced under $60.00.

ETO: Selects an early search result and the 52" by 90" option, then attempts a purchase.
Reward: 0.50 *Steps:* 4 *Outcome:* Failure

Ours: Visits multiple product pages, reasons about washability and price, selects the requested size, and attempts a purchase.
Reward: 0.75 *Steps:* 8 *Outcome:* Failure

Interpretation: Both trajectories remain unsuccessful. Additional attribute checks are present in the higher-reward summary, but they do not establish that every constraint was satisfied.

E.2 SCIENCEWORLD

Moving a Non-Living Object to the Green Box

Instruction: Find a non-living object, focus on it, and move it to the green box in the workshop.

ETO: Moves a yellow wire into the green box, then repeats waiting and inspection actions without an explicit focus step.
Reward: 0.25 *Steps:* 15 *Outcome:* Failure

Ours: Moves the same wire into the green box and issues focus actions on the box and the wire.
Reward: 0.75 *Steps:* 15 *Outcome:* Failure

Interpretation: The explicit focus actions distinguish the two attempts. The higher reward does not establish that all required actions were accepted or that the environment incorrectly withheld success.

Turning On a Green Light Bulb with Renewable Power

Instruction: Your task is to turn on the green light bulb. First, focus on the green light bulb. Then, create an electrical circuit that powers it on. Prefer renewable power sources when possible.

ETO: Explores circuit components and attempts connections, interspersed with focus actions on other objects.
Reward: 0.43 *Steps:* 30 *Outcome:* Failure

Ours: Focuses on the green bulb and connects wires involving the bulb and a renewable power component, repeatedly checking the state.
Reward: 0.58 *Steps:* 30 *Outcome:* Failure

Interpretation: The higher-reward summary contains a more task-directed sequence.

Growing a Banana from Seed to Fruit

Instruction: Your task is to grow a banana. This requires obtaining banana seeds, planting them in soil, providing water and light, and waiting until the banana grows.

ETO: Collects seeds and containers, with repeated navigation and inspection actions.
Reward: 0.36 *Steps:* 55 *Outcome:* Failure

Ours: Collects banana seeds and performs planting, watering, and growth-monitoring actions.
Reward: 0.50 *Steps:* 60 *Outcome:* Failure

Interpretation: The higher-reward failure includes several stages of the intended procedure. It does not establish that all growth conditions were met.

E.3 INTERCODESQL

Films Not Presented in China

Instruction: List the titles and directors of films that were never presented in China.

ETO: Inspects tables and repeatedly queries fields such as `Market` and `country` while attempting joins.
Reward: 0.00 *Steps:* 6 *Outcome:* Failure

Ours: Inspects schemas, revises table and column choices, and constructs a query through `market` for film titles and directors.
Reward: 0.77 *Steps:* 9 *Outcome:* Failure

Interpretation: The summary shows a revised query structure and higher partial reward. Executability and exact answer correctness must be distinguished; the reported result remains a failure.

Reviewers Who Rated Above 3 Stars

Instruction: Find the names of reviewers who previously rated a movie more than 3 stars.

ETO: Attempts joins across `reviewer`, `rating`, and `movie`, repeatedly revisiting their schemas.
Reward: 0.00 *Steps:* 10 *Outcome:* Failure

Ours: Inspects reviewer and rating fields and constructs a join using `rID` to retrieve reviewer names.
Reward: 0.75 *Steps:* 5 *Outcome:* Failure

Reviewers Who Rated Above 3 Stars (continued)

Interpretation: The revised join receives partial reward. The record does not justify calling the answer fully correct when the reported outcome is failure.

Gymnasts Ordered by Ascending Height

Instruction: Return the names of gymnasts ordered by their height in ascending order.

ETO: Attempts to retrieve names and heights from `gymnast`, with repeated schema inspection.
Reward: 0.00 *Steps:* 6 *Outcome:* Failure

Ours: Inspects both `gymnast` and `people`, then uses the name and height fields in a height-ordered query.
Reward: 0.70 *Steps:* 6 *Outcome:* Failure

Interpretation: The summary shows a change in table selection and ordering.

E.4 Training Failures and Target Behavior

The lemon-growing example compares unsuccessful training trajectories with subsequent target rollouts. The trajectory produced by the failure agent includes soil preparation and planting, while the baseline trajectory repeats a narrower set of actions. The target rollout produced by our method succeeds. The example documents this difference but does not show that a specific failure caused the successful target rollout.

Growing a Lemon with Cross-Pollination

Instruction: Your task is to grow a lemon. This will require growing several plants and having them cross-pollinated to produce fruit. Seeds can be found in the bedroom. To complete the task, focus on the grown lemon.

ETO: Target Rollout Retrieves the seed jar, plants seeds in water-filled pots, and repeats waiting, inspection, and focus attempts. The summary records no soil preparation.
Reward: 0.25 *Steps:* 60 *Outcome:* Failure
ETO: Training Failure Plants seeds in water-filled pots and repeatedly checks the greenhouse and attempts to focus on the lemon tree.
Reward: 0.25 *Steps:* 49 *Outcome:* Failure

Ours: Target Rollout Collects soil, fills the pots, plants seeds, waits for growth, closes greenhouse doors, and focuses on the lemon. The reported outcome is success.
Reward: 1.00 *Steps:* 46 *Outcome:* Success
Ours: Training Failure Collects and transports soil, fills pots, plants seeds, and later attempts focus and pickup actions referring to the lemon. The reported outcome remains failure.
Reward: 0.50 *Steps:* 60 *Outcome:* Failure
