CO-EVOLVING AGENTS: LEARNING FROM FAILURES AS HARD NEGATIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

The rapid progress of large foundation models has accelerated the development of task-specialized agents across diverse domains. However, the effectiveness of agents remains tightly coupled with the quality of training data, while curating task-specific datasets remains costly and often infeasible in real-world scenarios. Recent work has explored self-improving agents that autonomously generate, refine, and re-train on their own trajectories. A prominent line of approaches further leverages preference optimization by pairing predicted trajectories with scarce ground-truth trajectories, enabling agents to learn directly from their own failures. While these methods outperform supervised fine-tuning, their heavy reliance on predicted trajectories under limited ground-truth supervision leaves them prone to overfitting. To address this, we propose a *co-evolving agents framework* in which a target agent improves jointly with an auxiliary failure agent. The failure agent learns through preference optimization over failure trajectories from both the target and itself, thereby generating hard negatives that are close to success yet remain failures. Incorporating these informative hard negatives into the target agent's optimization sharpens decision boundaries and enhances generalization. Our comprehensive analysis and experiments across benchmark datasets show that our method not only show improved performance but also highlights that failures, instead of being used as-is, can be systematically transformed into structured and valuable learning signals in self-improving agents.

1 INTRODUCTION

The rapid progress of large foundation models (OpenAI, 2025; Yang et al., 2025; MetaAI, 2025; Anthropic, 2025; Gemini Team, 2025) has facilitated the rise of task-specialized agents across diverse domains, ranging from open-domain dialogue to scientific reasoning tasks (SU et al., 2025; Zeng et al., 2024a; Fu et al., 2025; Bousmalis et al., 2024). These agents inherit the broad generalization capacity of pretrained models, allowing effective adaptation to new tasks with relatively limited supervision. This promise has motivated growing interest in developing methods that adapt foundation models into reliable and effective domain-specialized agents. Recent advances in multi-agent systems and preference optimization further highlight the potential of combining broad pretraining with specialized adaptation. In particular, the ability to automatically curate training signals from agent interactions opens opportunities for scaling beyond static, human-labeled corpora. At the same time, the increasing deployment of agents in dynamic, real-world settings emphasizes the importance of approaches that can continuously refine behavior without costly retraining. Such developments make it timely to revisit how failures, long viewed as undesirable artifacts, can instead be leveraged as constructive learning signals.

Nevertheless, the effectiveness of such agents remains constrained by the quality of task-specific training data (Zhou et al., 2024; Zhao et al., 2024). High-quality datasets are essential for reliable reasoning and decision making, providing the signals required for adaptation to specialized domains. Yet, constructing such datasets is expensive and labor-intensive, often requiring domain expertise and extensive annotation. In many real-world scenarios, constructing large curated datasets is infeasible. In addition to the prohibitive cost of collecting interactions at scale, the need to repeatedly curate data to keep pace with non-stationary environments makes this approach impractical. This bottleneck has motivated growing interest in methods that enable agents to improve autonomously without relying on continuous manual curation (Yuan et al., 2025b; Nguyen et al., 2025; Yin et al.,

2025). Such methods aim to bridge the gap between the flexibility of foundation models and the rigorous demands of domain-specific tasks, while keeping human intervention to a minimum. A central challenge is therefore to design mechanisms that transform abundant but noisy interaction data into structured supervision that drives reliable improvement.

Self-improving agents (SU et al., 2025; Zeng et al., 2024a; Fu et al., 2025) have emerged as a promising paradigm to reduce reliance on costly human annotation. Maintaining agents at state-of-the-art performance would require continuous human annotation, which is prohibitively costly and infeasible at scale. Instead, self-improving agents automate parts of the data construction process by synthesizing expert-like trajectories from external resources such as documentation or databases, and by repurposing predicted failures as preference data for training.

Building on this idea, Exploration-Based Trajectory Optimization (ETO) (Song et al., 2024) constructs preference datasets by pairing agent-generated failures with ground-truth trajectories using a given reward model, enabling preference optimization (Rafailov et al., 2023). Despite the promise of autonomous improvement, these approaches remain limited, as they depend on a small set of ground-truth successes paired with predicted failures, leaving them prone to overfitting.

To overcome these limitations, we introduce a *co-evolving agents framework* in which a target agent improves jointly with an auxiliary failure agent. The failure agent specializes in preference optimization over failure trajectories from both the target and itself, enabling it to learn a fine-grained landscape of failures rather than merely preferring expert trajectories. By doing so, it generates hard negatives (Robinson et al., 2021; Rafailov et al., 2023; Chen et al., 2020), which are failures close to success, and these provide stronger and more diverse contrastive signals. Incorporating these informative hard negatives into the target agent's preference optimization sharpens decision boundaries and yields more generalizable performance.

We validate our framework through comprehensive analysis and experiments across diverse domains, including the online shopping environment WebShop (Yao et al., 2022), the science reasoning environment ScienceWorld (Wang et al., 2022), and the interactive SQL environment InterCodeSQL (Yang et al., 2023). Our analysis on these benchmarks verifies that the failure agent does not simply imitate expert trajectories but continues to generate high-reward failures that serve as informative hard negatives.

Experiments further demonstrate substantial improvements over competitive baselines, achieving large margins of gain across benchmarks and reflecting stronger generalization to diverse tasks. These findings highlight that systematically harnessing failures as structured learning signals, rather than treating them as byproducts, opens a promising direction for advancing self-improving agents.

Our contributions are summarized as follows: double-check

- We introduce a *failure agent* that, unlike prior frozen negative agents trained on human-curated data, continuously learns from failure trajectories and captures a fine-grained failure landscape.
- We propose a *co-evolving agents framework* where a target and failure agent improve jointly, with the failure agent generating hard negatives that sharpen decision boundaries and enhance generalization.
- Our experiments further confirm that failures, when systematically harnessed, can be transformed into structured learning signals that drive more robust self-improving agents.

2 Related work

Self-Improving Agents Building high-performing agents requires high-quality datasets, which are costly and often infeasible in real-world scenarios. Self-improving agents address this by autonomously generating, refining, and reusing data for continual learning. Some approaches synthesize trajectories from tutorials, documentation, or persona hubs (SU et al., 2025; Zeng et al., 2024a; Fu et al., 2025), while others use planning methods such as Monte Carlo Tree Search (MCTS) (Yuan et al., 2025b). Beyond dialogue, self-improvement has been explored in programmatic action composition (Nguyen et al., 2025), robotics (Bousmalis et al., 2024), and code generation (Yin et al., 2025). Another line leverages failure trajectories paired with expert ones for preference optimization (Song et al., 2024; Xiong et al., 2024), but these typically use failures as-is, limiting generalization. Multi-agent variants (Zhang et al., 2024) employ negative agents trained on curated failure

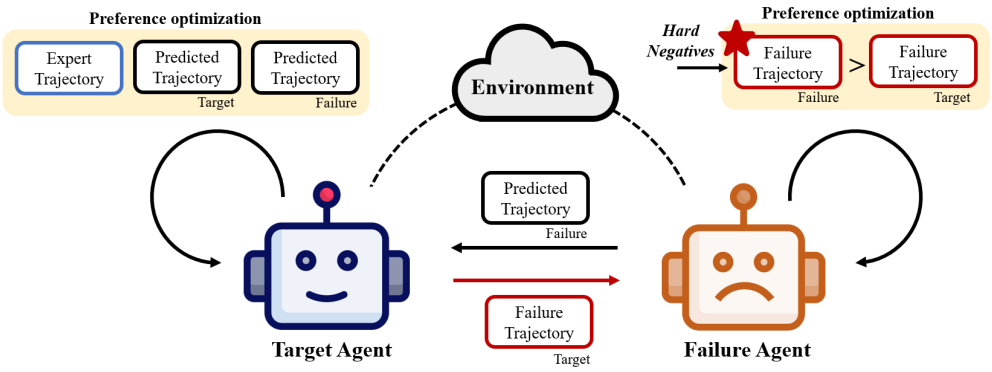


Figure 1: Overview of our co-evolving agents framework.

datasets, yet these are frozen and restricted to dialogue tasks, offering only limited benefit compared to success-based supervision.

Hard Negatives in Contrastive Optimization Reinforcement Learning from Human Feedback (RLHF) (Lee et al., 2024) has been the standard paradigm for aligning language models, but it requires costly reward modeling and policy optimization. Recent contrastive methods such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) and Generalized Preference Optimization (GRPO) (Tang et al., 2024) simplify this process by directly optimizing policies on preference pairs, bypassing explicit reward models. At the core of contrastive optimization is the idea that learning benefits most from informative comparisons. In particular, *hard negatives* that are difficult to distinguish from the preferred ones and thus yield small preference margins, are known to provide stronger supervision and promote sharper decision boundaries (Robinson et al., 2021; Rafailov et al., 2023; Chen et al., 2020).

3 Preliminaries

The interaction between an LLM agent and its environment can be formalized as a partially observable Markov decision process (POMDP) () $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, T, R)$, as in Song et al. (2024). Here, $\mathcal{U}$ denotes the instruction space, $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition function, and $R : \mathcal{S} \times \mathcal{A} \to [0, 1]$ the reward function. In our LLM-agents setting, $\mathcal{U}, \mathcal{A}$, and $\mathcal{O}$ are expressed in natural language.

At the beginning of each episode, the agent receives an instruction $u \in \mathcal{U}$ and generates its first action $a_1 \sim \pi_\theta(\cdot \mid u) \in \mathcal{A}$ from its policy π_θ parameterized by θ. The action updates the latent state $s_t \in \mathcal{S}$ and produces an observation $o_t \in \mathcal{O}$. Subsequent actions are conditioned on the full interaction history, so that

$$a_t \sim \pi_\theta(\cdot \mid u, a_1, o_1, \ldots, a_{t-1}, o_{t-1}) \in \mathcal{A}.$$

This process unfolds until either the task is solved or the step budget is exceeded. A trajectory can therefore be written as

$$e = (u, a_1, o_1, \ldots, o_{n-1}, a_n) \sim \pi_\theta(e \mid u), \tag{1}$$

with likelihood

$$\pi_\theta(e \mid u) = \prod_{j=1}^{n} \pi_\theta(a_j \mid u, a_1, o_1, \ldots, o_{j-1}), \tag{2}$$

where n is the trajectory length.

Finally, a reward $r(u, e) \in [0, 1]$ is assigned to the trajectory, where $r(u, e) = 1$ corresponds to full task success and lower values indicate partial or failed attempts. This formulation sets up the basis for preference-based training methods that compare trajectories according to their rewards.

4 METHOD

In this section, we propose our *co-evolving agents framework* in which a target agent and a failure agent are trained in alternating phases and gradually improve through mutual interaction. In Section 4.1, we describe the behavioral cloning stage used to initialize the base policy from supervised fine-tuning on expert trajectories. Next, in Section 4.2, we introduce the failure agent, which learns via preference optimization over failure trajectories from both the target and itself and generates fine-grained hard negatives that are close to success yet still failures. Finally, in Section 4.3, we describe how the target agent leverages expert trajectories, its own predicted failures, and failures generated by the failure agent to construct diverse preference datasets for direct preference optimization (DPO) (Rafailov et al., 2023). By training on this richer set of comparisons and alternating with the failure agent, the target agent achieves more effective learning and stronger generalization within a co-evolutionary loop. The overall pipeline is illustrated in Figure 1.

4.1 BEHAVIORAL CLONING WITH SUPERVISED FINE-TUNING

We first initialize a base policy through behavioral cloning, which equips the agent with fundamental task-solving ability before self-improvement. Given an expert dataset $\mathcal{D} = \{(u^{(i)}, e^{(i)})\}_{i=1}^{|\mathcal{D}|}$, each trajectory $e = (u, a_1, o_1, \ldots, a_n)$ consists of a task instruction u, actions $a_t \in \mathcal{A}$, and observations $o_t \in \mathcal{O}$. The agent policy π_θ is trained with an autoregressive supervised fine-tuning (SFT) objective:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}_{e\sim\mathcal{D}}\big[\log \pi_\theta(e \mid u)\big], \tag{3}$$

where the trajectory likelihood decomposes as

$$\pi_\theta(e \mid u) = \prod_{t=1}^{n} \pi_\theta(a_t \mid u, a_{<t}, o_{<t}). \tag{4}$$

In practice, the instruction, actions, and observations are concatenated into a single text sequence $t = (t_1, t_2, \ldots, t_l)$. The loss is then computed by applying the autoregressive likelihood only to tokens corresponding to agent actions:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_{k=1}^{l} \log \pi_\theta(t_k \mid t_{<k}) \cdot \mathbf{1}(t_k \in \mathcal{A}), \tag{5}$$

where $\mathbf{1}(t_k \in \mathcal{A})$ is an indicator that selects tokens generated as agent actions.

This supervised fine-tuning stage provides the base policy π_{base}, which serves as the starting point for co-evolution with the failure agent. To ensure simplicity, both target and failure agents are initialized from independently trained base policies on the same expert dataset, providing a comparable starting point while allowing only minor stochastic differences.

4.2 FAILURE AGENT FOR GENERATING HARD NEGATIVES

We introduce an auxiliary *failure agent* π_{θ_f} whose role is to specialize in unsuccessful trajectories and refine them into informative hard negatives (). Unlike the target agent π_{θ_t}, which is optimized toward expert success, the failure agent focuses on modeling the space of failures and extracting fine-grained signals from them. This complementary specialization enables the two agents to co-evolve in alternating phases.

Preference Dataset. The preference dataset for the failure agent consists of failure trajectories generated by both the target and itself. Formally, let $\mathcal{F}_{\text{tgt}} = \{e_{\text{tgt}} \mid r(u, e_{\text{tgt}}) < 1\}$ and $\mathcal{F}_{\text{fail}} = \{e_{\text{fail}} \mid r(u, e_{\text{fail}}) < 1\}$ denote the sets of failure trajectories generated by the target and failure agents, respectively. We construct a preference dataset by pairing failures with different reward levels:

$$\mathcal{D}_{\text{fail}} = \big\{(u, e_{\text{chosen}}, e_{\text{rejected}}) \,\big|\, e_{\text{chosen}}, e_{\text{rejected}} \in \mathcal{F}_{\text{tgt}} \times \mathcal{F}_{\text{fail}},\ r(u, e_{\text{chosen}}) > r(u, e_{\text{rejected}})\big\}. \tag{6}$$

Here, both e_{chosen} and e_{rejected} are failure trajectories, and their relative preference is determined by evaluating rewards with the given reward model, where the higher-reward trajectory is selected as e_{chosen} and the lower as e_{rejected}. This construction enables the failure agent to exploit not only its own generated failures but also those produced by the target, providing a richer set of pairwise comparisons.

Preference Optimization. We adopt the direct preference optimization (DPO) objective (Rafailov et al., 2023) for training on failure trajectories. Given a reference policy π_{ref}, the failure agent π_{θ_f} is updated by

$$\mathcal{L}_{\text{DPO}}(\theta_f) = -\mathbb{E}_{(u,e_{\text{chosen}},e_{\text{rejected}})\sim\mathcal{D}_{\text{fail}}}\Big[\log\sigma\big(\beta\log\frac{\pi_{\theta_f}(e_{\text{chosen}}|u)}{\pi_{\text{ref}}(e_{\text{chosen}}|u)} - \beta\log\frac{\pi_{\theta_f}(e_{\text{rejected}}|u)}{\pi_{\text{ref}}(e_{\text{rejected}}|u)}\big)\Big], \tag{7}$$

where u is the task instruction, $\sigma(\cdot)$ is the logistic sigmoid, and β is a scaling factor. This objective drives the failure agent to distinguish between relatively better and worse failures, thereby capturing subtle distinctions within the failure space.

Hard Negatives. By constructing preference datasets over failure–failure pairs and training with the DPO objective, the failure agent learns to capture nuanced differences among failures rather than only aligning with expert trajectories. This enables fine-grained learning of the failure trajectory landscape by considering diverse failure cases and, in particular, generating near-success failures that remain informative despite not solving the task. Such hard negatives provide informative signals that cannot be obtained from simple expert-versus-failure comparisons, allowing the model to establish a sharper decision boundary between success and failure. As a result, we incorporate these refined failure trajectories into the target agent's optimization, leading to improved robustness and stronger generalization. To better understand the role of the failure agent, we further conduct both quantitative and qualitative analyses of the generated failure trajectories (Section 5.2.1).

4.3 Co-evolutionary Training

The target agent improves by incorporating failure trajectories generated by the failure agent into its preference optimization. We construct a preference dataset $\mathcal{D}_{\text{tgt}}$ consisting of three types of trajectory pairs: (i) expert trajectories versus target-predicted trajectories, (ii) expert trajectories versus failure-agent trajectories, and (iii) failure trajectories from the target versus those from the failure agent. Formally,

$$\mathcal{D}_{\text{tgt}} = \Big\{(u, e_{\text{chosen}}, e_{\text{rejected}}) \;\Big|\; (e_{\text{chosen}}, e_{\text{rejected}}) \in \{(e_{\text{exp}}, e_{\text{tgt}}), (e_{\text{exp}}, e_{\text{fail}})\} \cup (\mathcal{F}_{\text{tgt}} \times \mathcal{F}_{\text{fail}})\Big\}, \tag{8}$$

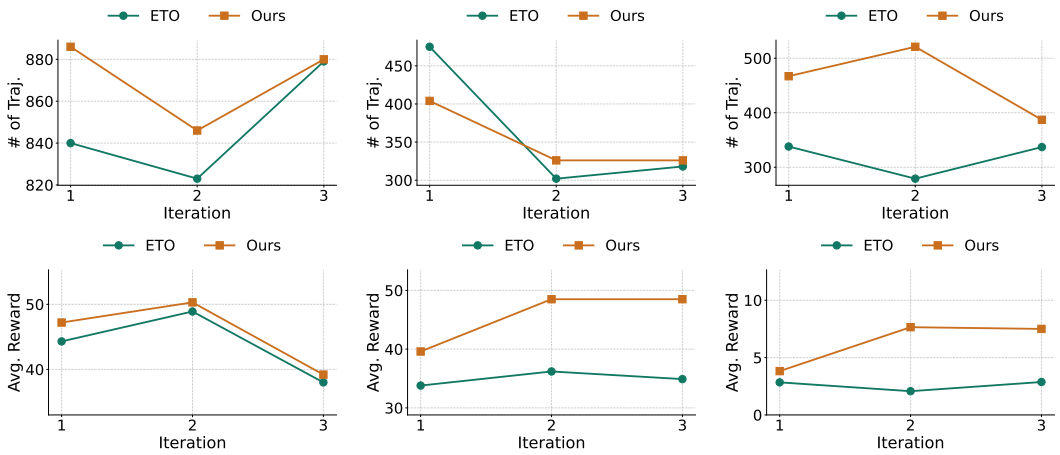
where e_{exp} denotes expert trajectories, e_{tgt} target-predicted trajectories, and e_{fail} failure-agent trajectories. Here, $\mathcal{F}_{\text{tgt}} = \{e_{\text{tgt}} \mid r(u, e_{\text{tgt}}) < 1\}$ and $\mathcal{F}_{\text{fail}} = \{e_{\text{fail}} \mid r(u, e_{\text{fail}}) < 1\}$ denote the sets of failed trajectories generated by the target and failure agents, respectively.

The target agent is optimized with a weighted DPO objective (Rafailov et al., 2023) together with an auxiliary supervised fine-tuning (SFT) loss on the chosen trajectories:

$$\mathcal{L}_{\text{target}} = \lambda_{\text{DPO}}\,\mathcal{L}_{\text{DPO}} + \lambda_{\text{SFT}}\,\mathbb{E}_{(u,e_{\text{chosen}})\sim\mathcal{D}_{\text{tgt}}}\big[-\log\pi_\theta(e_{\text{chosen}} \mid u)\big]. \tag{9}$$

As noted by Yuan et al. (2025a), DPO alone maximizes relative preference margins but can become unstable, since the space of chosen trajectories is much smaller than that of rejected ones. This imbalance may lead the model to over-penalize rejected samples while insufficiently reinforcing preferred ones. To stabilize training, we introduce the auxiliary SFT loss on the chosen trajectories, which grounds the policy toward high-reward behaviors. For expert–prediction pairs, we set $(\lambda_{\text{DPO}}, \lambda_{\text{SFT}}) = (0.5, 0.5)$ so that the total contribution approximates the conventional expert–prediction preference setup. For failure–failure pairs, we use $(\lambda_{\text{DPO}}, \lambda_{\text{SFT}}) = (1.0, 0.0)$, relying purely on preference optimization to avoid confusing supervision from incorrect trajectories.

This design prevents over-counting of expert signals while ensuring that failure pairs receive full emphasis. Failure–failure comparisons are especially valuable, as they capture subtle distinctions between suboptimal behaviors, thereby sharpening the agent's decision boundaries. As training alternates with the failure agent, the two agents form an implicit arms race: the failure agent generates increasingly challenging negatives, while the target agent learns to overcome them. This co-evolutionary loop not only enhances robustness within the training domain but also improves generalization to unseen environments where expert supervision is limited.



(a) WebShop (b) ScienceWorld (c) InterCodeSQL

Figure 2: Analysis of Failure Trajectory

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Datasets We conduct experiments on three representative benchmarks: *WebShop* for web navigation, *ScienceWorld* for scientific reasoning, and *InterCodeSQL* for interactive SQL querying. All three environments provide continuous final rewards in $[0, 1]$, enabling fine-grained evaluation of task completion. Expert trajectories are collected through a combination of human annotations and GPT-4–assisted generation in the ReAct format (Yao et al., 2023), with additional filtering based on final rewards to ensure quality. Example trajectory samples for each dataset are provided in Figures 4 to 6. Further details of the environments and trajectory collection process are provided in Appendix A.

Implementation Details We adopt Llama-2-7B-Chat (Touvron et al., 2023) as the primary base model, following prior work (Song et al., 2024). All models are optimized with AdamW (Loshchilov & Hutter, 2017), and each training phase is performed for 3 epochs with co-evolution iterations set to 3 for WebShop and ScienceWorld and 5 for InterCodeSQL. All other hyperparameters are kept identical across datasets to ensure fair comparison. Experiments are conducted on 8 NVIDIA H100 GPUs with 80GB memory, and further implementation details are provided in Appendix B.

Baselines We compare our framework with standard imitation learning and several strong post-imitation baselines following the baseline Song et al. (2024). Supervised fine-tuning (SFT) (Chen et al., 2023; Zeng et al., 2024b) trains agents via behavioral cloning on expert trajectories and serves as the base policy for other methods. Rejection Fine-Tuning (RFT) (Yuan et al., 2023) augments the expert dataset with success trajectories identified by rejection sampling, while Proximal Policy Optimization (PPO) (Schulman et al., 2017) directly optimizes the SFT policy with reinforcement learning to maximize task rewards. For reference, we also report results from GPT-3.5-Turbo (Ouyang et al., 2022), GPT-4 (Achiam et al., 2023) with in-context learning. We report average reward as the primary evaluation metric.

5.2 RESULTS

5.2.1 ANALYSIS ON FAILURE TRAJECTORIES

To better understand the role of the failure agent, we conduct a quantitative and qualitative analysis of the generated failure trajectories.

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	59.2	47.3	41.9	30.8	44.8
	Llama-2-7B-Chat + PPO	64.2	59.4	51.7	52.4	56.9
	Llama-2-7B-Chat + RFT	61.3	71.6	54.3	35.6	55.7
	Llama-2-7B-Chat + ETO	63.0	65.6	55.5	**51.7**	59.0
	Llama-2-7B-Chat + Ours	**68.5**	**72.0**	**66.3**	49.6	**64.1**

Table 1: Main results.

Quantitative Analysis For this analysis, we exclude instructions that are already solved during supervised fine-tuning and focus on the remaining failure cases. We then compare ETO and our method in terms of (i) the total number of failure trajectories produced and (ii) the average reward of failures shared by both methods. As shown in Figure 2, our method not only generates a larger pool of failures but also produces failures with higher rewards on overlapping samples, which correspond to more informative *hard negatives* that are closer to success yet still unsuccessful. This confirms that the failure agent operates as intended, generating more informative near-success failures that serve as valuable hard negatives for training.

Qualitative Analysis To better understand the role of failure trajectories, we qualitatively compare trajectories generated by ETO and by our method on the *ScienceWorld*, using the task of boiling a marshmallow as an illustrative example. As shown in the box below, ETO produces a degenerate failure: after issuing invalid actions that prevent the marshmallow from being placed in the pot, the agent falls into a loop of repeatedly inspecting the stove, ultimately reaching the step limit with negligible reward. More generally, ETO trajectories tend to collapse into such trivial loops or, conversely, achieve perfect 1.0 success, leaving few informative cases in between. In contrast, our method generates more structured failure trajectories: the agent navigates to the kitchen, retrieves a pot, places the marshmallow inside, and activates the stove. Although not always successful, these attempts achieve substantially higher reward and capture essential sub-skills such as navigation, object manipulation, and device control.

Such near-successful attempts correspond to *hard negatives*—failures that are close to the correct solution and therefore more informative for preference learning than either degenerate failures or trivial successes. Because our framework explicitly emphasizes these hard negatives, the resulting target agent is trained on richer supervision, which in turn explains the consistent improvements over conventional expert–prediction baselines.

Predicted Trajectories

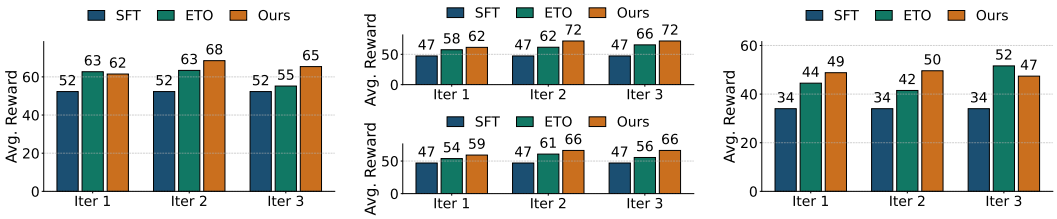
Instruction: Your task is to boil marshmallow. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.

ETO (baseline): Attempts to combine marshmallow and pot fail due to invalid actions. The pot remains empty, after which the agent repeatedly issues "`look at stove`" without progress until the step limit. *Reward:* 0.03 *Steps:* 100 *Outcome:* Failure.

Ours: The agent retrieves a pot, correctly moves the marshmallow inside, places it on the stove, and activates heating. Subsequent checks show the marshmallow turning into liquid, completing the task. *Reward:* 0.75 *Steps:* 43 *Outcome:* Success.

5.2.2 Main Results

We evaluate our framework on three representative benchmarks: WebShop for web navigation, ScienceWorld for scientific reasoning, and InterCodeSQL for interactive SQL querying. Each bench-



(a) WebShop (b) ScienceWorld (Seen / Unseen) (c) InterCodeSQL

Figure 3: Average reward across iterations.

mark requires multi-step decision making with final rewards in $[0, 1]$, providing a fine-grained measure of task success.

Table 1 reports the results. Under in-context learning without fine-tuning, GPT-4 and GPT-3.5-Turbo achieve 63.2 and 62.4 on WebShop, but their performance collapses to 42.9 and 7.9 on ScienceWorld, highlighting the limitations of prompt-only adaptation in reasoning-intensive or previously unseen domains. Fine-tuning methods substantially boost performance by aligning models with domain-specific interaction patterns. Among them, ETO serves as a strong baseline, reaching an average reward of 59.0. Please note that the results of PPO and RFT on ScienceWorld are taken from Song et al. (2024); Xiong et al. (2024)

Our method achieves an average reward of 64.1, outperforming ETO by +5.1 points. The improvements are consistent across benchmarks: +5.5 on WebShop, +6.4 on seen ScienceWorld, and +10.8 on unseen ScienceWorld. In contrast, performance on InterCodeSQL remains on par with the baseline. A closer analysis indicates that InterCodeSQL exhibits an extremely sparse reward structure, where trajectories are almost always scored as either 0 or 1. This binary feedback limits the construction of informative failure pairs, reducing the advantage of our framework and leading both methods to converge to similar outcomes. Notably, the largest gain arises on *unseen ScienceWorld tasks*, where our approach surpasses ETO by more than 10.8 points, demonstrating significantly stronger generalization to novel scientific scenarios. Overall, these results highlight that leveraging failure-agent trajectories to generate informative hard negatives provides richer training signals, yielding both higher robustness and improved out-of-distribution generalization.

5.3 Ablation Study

We conduct ablation studies to better understand the contribution of each component in our framework. First, we analyze performance across co-evolution iterations (Figure 3). Our method achieves strong improvements even with fewer iterations, suggesting that the failure agent generates more effective trajectories that accelerate learning compared to standard exploration.

Additionally, on WebShop, we evaluate whether the failure agent is genuinely beneficial by replacing it with a conventional positive agent that, like ETO, learns solely from expert-versus-predicted comparisons. This variant achieves an average reward of 62.76, which is almost identical to ETO, yet notably lower than the 68.5 obtained with the failure agent, confirming that explicitly modeling and refining failures is more effective than simply ensembling additional success-oriented agents.

6 Conclusion

We introduced a co-evolving agents framework where a target agent and a failure agent learn in alternating phases and improve through mutual interaction. By training the failure agent on failure–failure preferences, it generates near-success failures that serve as informative hard negatives. Incorporating these trajectories into the target agent's preference optimization sharpens decision boundaries and improves robustness and generalization. Experiments across WebShop, ScienceWorld, and InterCodeSQL demonstrate consistent gains in diverse domains, underscoring that failures, when refined into structured signals, can be transformed into valuable resources for self-improving agents. We hope our findings facilitate more principled handling of failure trajectories, ultimately contributing to the advancement of the next generation of self-improving agents.

REFERENCES

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

Anthropic. Introducing claude 4. https://www.anthropic.com/news/claude-4, 2025. Accessed: 2025-09-24.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Manon Devin, Alex X. Lee, Maria Bauza Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Fernandes Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Zolna, Scott Reed, Sergio Gómez Colmenarejo, Jonathan Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Thomas Rothörl, Jose Enrique Chen, Yusuf Aytar, David Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=vsCpILiWHu.

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.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma GongQue, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. Agentrefine: Enhancing agent generalization through refinement tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FDimWzmcWn.

Google DeepMind Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical Report Report, 16 June 2025, Google DeepMind, 2025. Accessed: 2025-09-24.

Harrison Lee, Sam Phatale, August Pritzel, Véronique Dalibard, Maria Tsimpoukelli, Dale Guo, S. M. Amin Hosseini, Kimin Lee, Amelia Glaese, Geoffrey Irving, and James D. Cohen. RLAIF: Reinforcement learning from ai feedback. In *International Conference on Machine Learning*, 2024. Based on arXiv:2309.00267.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

MetaAI. The llama 4 herd: Multimodal intelligence. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025. Accessed: 2025-09-24.

Dang Nguyen, Viet Dac Lai, Seunghyun Yoon, Ryan A. Rossi, Handong Zhao, Ruiyi Zhang, Puneet Mathur, Nedim Lipka, Yu Wang, Trung Bui, Franck Dernoncourt, and Tianyi Zhou. Dynasaur: Large language agents beyond predefined actions. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=lv0cJ2pWVd.

OpenAI. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/, 2025. Accessed: 2025-09-24.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

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. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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/.

Hongjin SU, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan O Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=3UKOzGWCVY.

Yunhao Tang, Zhaohan Daniel Guo, Zeyu Zheng, Daniele Calandriello, Remi Munos, Mark Rowland, Pierre Harvey Richemond, Michal Valko, Bernardo Avila Pires, and Bilal Piot. Generalized preference optimization: A unified approach to offline alignment. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=gu3nacA9AH.

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/.

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, Fan Yang, 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 R Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=fvKaLF1ns8.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

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 (ICLR)*, 2023.

Xunjian Yin, Xinyi Wang, Liangming Pan, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential framework helps for recursively self-improvement, 2025. URL https://openreview.net/forum?id=dML3XGvWmy.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. Advancing LLM reasoning generalists with preference trees. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=2ea5TNVR0c.

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*, 2025b.

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, 2024a.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Yuxuan Zhang, Ruida Wang, Z. Quantin, Yifei Li, Xuefei Ning, Shengyi Jiang, Lijun Wu, Shuaicheng Niu, Yuping Wang, Yu Sun, Kegong Tian, and Xiang Li. MACPO: Weak-to-strong alignment via multi-agent contrastive preference optimization. In *International Conference on Learning Representations*, 2024. arXiv:2410.07672.

Wanru Zhao, Hongxiang Fan, Shell Xu Hu, Wangchunshu Zhou, and Nicholas Lane. Clues: Collaborative private-domain high-quality data selection for llms via training dynamics. *Advances in Neural Information Processing Systems*, 37:141183–141207, 2024.

Hang Zhou, Yehui Tang, Haochen Qin, Yujie Yang, Renren Jin, Deyi Xiong, Kai Han, and Yunhe Wang. Star-agents: Automatic data optimization with llm agents for instruction tuning. *Advances in Neural Information Processing Systems*, 37:4575–4597, 2024.