FRONTIER-RSI: ADVANCING THE AGENT CAPABILITY FRONTIER THROUGH CONTINUAL TRAINING AND ENVIRONMENT EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement (RSI) seeks to enable agents to continually expand their capabilities through iterative learning and evolution. However, existing approaches largely treat evolution as an open-loop process: as the agent improves, the evolving training environments are not systematically adapted to its changing capabilities, leaving no closed loop that continually pushes the agent's capability frontier. We introduce FRONTIER-RSI, a closed-loop framework that couples agent continual learning, policy-relative task states, and environment evolution. After each learning round, FRONTIER-RSI measures each task relative to the updated agent and categorizes it as unsolved, frontier, or saturated. A curriculum operator then adapts the environment through two complementary knobs, difficulty and scaffolding, to continuously reshape the task distribution around the agent's learnable frontier. Specifically, saturated tasks that have become too easy are evolved into harder escalation tasks, while unsolved tasks that remain beyond the agent's reach are transformed into learnable prefix tasks by providing verified intermediate progress. Experiments across foundation models with different capability levels and coding tasks of varying difficulty demonstrate that FRONTIER-RSI consistently sustains improvement across successive rounds, effectively advancing the agent's capability frontier. Code is open-sourced at https://anonymous.4open.science/r/Frontier-RSI-1C0D/.

1 INTRODUCTION

Recursive self-improvement (RSI) offers a promising direction for advancing AI by enabling AI systems to take an increasing role in their own development (Chen et al., 2026a;b). Its central promise is a cumulative improvement process: a system uses its current capabilities to discover, evaluate, and incorporate improvements (*e.g., post-training* (Wei et al., 2026; Wang et al., 2026a)), and the resulting capabilities support subsequent rounds of improvement. By automating parts of this cycle, RSI could reduce dependence on repeated human intervention in data curation, system design, and adaptation, providing a scalable path toward increasingly capable and autonomous agents.

Existing approaches pursue RSI through three complementary routes. Harness-level methods optimize the scaffolding around a fixed base policy, including prompts, tool interfaces, and execution control, through external search, trace-driven self-modification, or learned controllers (Lee et al., 2026; Zhang et al., 2026; Wang et al., 2026c). These changes improve agent behavior but do not directly consolidate new capabilities into model parameters. Environment-level methods expand the supply and complexity of executable training tasks (Ivison et al., 2026; Shen et al., 2026; Li et al., 2026b), yet recursive generation can remain open-loop with respect to the learner: verifying that a task is solvable does not establish whether the current policy can learn from it. Model-level methods couples task generation with policy learning (Zhao et al., 2025; Huang et al., 2025; Wei et al., 2025), but sustaining a useful curriculum remains challenging. Solver feedback can be weak or noisy, and proposer incentives can favor narrow families of challenges over broader capability growth (Wei et al., 2025) without external information.

Building on these advances, we conceptualize RSI through three questions: Recursive asks whether the environment can continually evolve; Improvement asks whether the agent can continually learn and advance its capability frontier; and Self asks whether this closed loop can operate autonomously. The central question then becomes:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

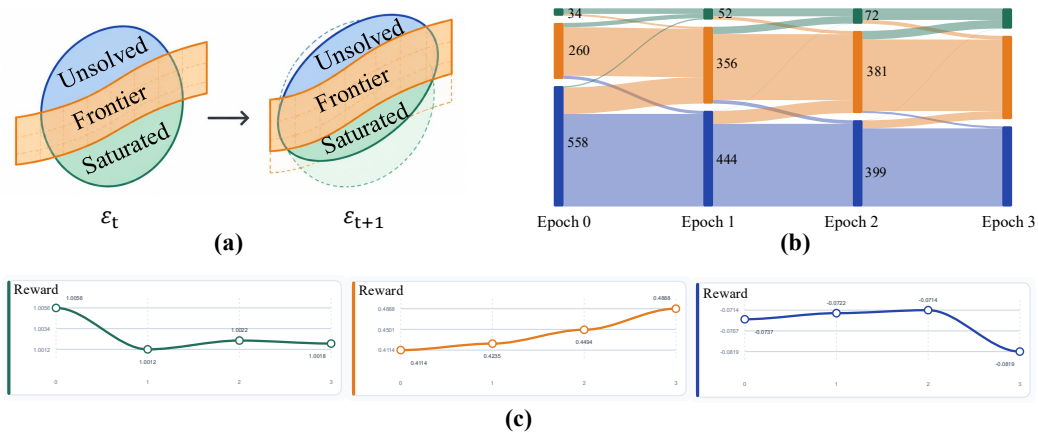

Figure 1: FRONTIER-RSI in principle and in practice. **(a)** Evolving the environment from $\mathcal{E}_t$ to $\mathcal{E}_{t+1}$ re-partitions the task pool, so the frontier band moves onto tasks that are unsolved instead of staying where the agent has already saturated. **(b)** The task states flow measured while training `Qwen3-8B` on TMax: the flow of the 852 seed tasks between the three task states across training epochs. **(c)** Mean reward within each task state over the same epochs.

Can task states measured after each training round automatically guide environment evolution, and continually advance the agent's capability frontier?

To investigate this question, we introduce FRONTIER-RSI, a framework that couples agent continual learning, task states, and environment evolution. After each training round, we evaluate tasks with the updated policy and categorize them as unsolved, frontier, or saturated. These states characterize task solvability relative to the current agent under a fixed evaluation configuration. A curriculum operator then uses this feedback to reshape the next training pool through two complementary control knobs: *difficulty* and *scaffolding*. Repeating this process closes the loop from policy improvement to task-state measurement, environment adaptation, and further learning. Figure 1(a) provides a schematic overview of our method.

Specifically, the curriculum operator applies different transformations according to each task's state. Figure 1(b) illustrates how tasks transition between states during training, providing empirical support for our divide-and-conquer approach. For unsolved tasks, it constructs *prefix tasks* by materializing verified intermediate progress as the initial environment state. The agent completes the remaining work under the original instruction and completion criteria. For saturated tasks, it constructs *escalation tasks* by introducing recoverable workspace obstacles or strengthening public requirements. For frontier tasks, it refreshes the available challenges through consistent rewriting of the instruction, reference solution, and verifier.

We evaluate FRONTIER-RSI with two foundation models using terminal and software-engineering tasks. Across the reported settings, improvements continue beyond the initial training round. Extensive experiments are also implemented for the ablation study and detailed analysis. These results support continued capability improvement over the evaluated rounds and transfer to tasks outside the evolving training pool.

Our contributions are threefold:

- We introduce FRONTIER-RSI, a closed-loop framework connecting agent continual learning and environment evolution through task states.
- We develop a curriculum operator combining verified prefix scaffolding, task escalation, and frontier-task rewriting to evolve environments.
- Our training recipe is also open-sourced for pushing the research of RSI forward.

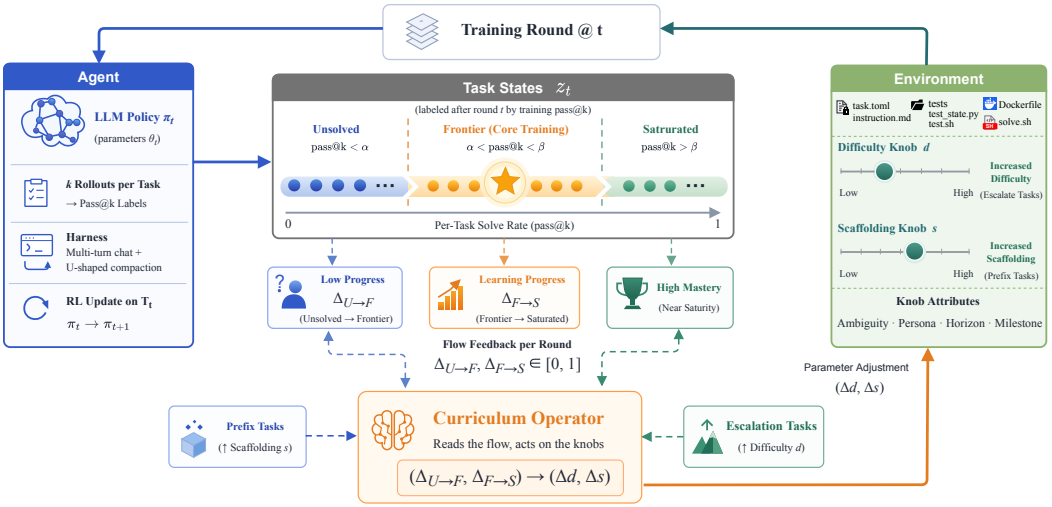

Figure 2: Framework overview of FRONTIER-RSI. The loop closes as agent continual learning → task states → environment evolution.

2 RELATED WORK

2.1 HARNESS-LEVEL RSI

An agent's behavior depends on both its underlying model and the harness that governs its execution. META-HARNESS (Lee et al., 2026) uses an external agentic proposer to search over harness source code, drawing on candidate implementations, evaluation scores, and execution traces. Complementing these approaches, HARNESSBRIDGE (Wang et al., 2026c) trains a plug-in controller to compress interaction histories into decision-relevant states and translate proposed actions into executable transitions or grounded rejections. Recently, DREAM-RSI (Zheng et al., 2026) helps agents learn to make smarter decisions about what to try next by repeatedly replaying past trial-and-error experiences.

2.2 MODEL-LEVEL RSI

This roadmap uses model-generated supervision and interaction feedback to update the underlying policy model. R-ZERO and SOCRATIC-ZERO develop adaptive curricula through proposer-solver co-evolution (Huang et al., 2025; Wang et al., 2025), while ABSOLUTE-ZERO jointly develops task-proposing and task-solving capabilities through execution-verified self-play (Zhao et al., 2025). SPIRAL derives learning signals from competitive self-play (Liu et al., 2025), and SOCRATIC-GEO and SPATIALEVO investigate environment-grounded self-evolution in geometric and spatial reasoning (Jiao et al., 2026; Li et al., 2026a). SSR (Wei et al., 2025) uses a single LLM agent to iteratively inject and repair software bugs of increasing complexity. SOCRATIC-SWE distills solving traces into skills, generates targeted repair tasks, and selects verified candidates to update solvers for subsequent rounds (Xiao et al., 2026).

2.3 ENVIRONMENT-LEVEL RSI

It evolves by recursively synthesizing environments at scale to generate more training data. TMAX scales environment generation through compositional control of task complexity, personas, and verifiers (Ivison et al., 2026). SETA (Shen et al., 2026) is a scalable framework for generating verifiable terminal environments for RL. RST (Li et al., 2026b) further develops recursive task synthesis by extending reference solutions, realigning instructions and verifiers, and reusing validated tasks as seeds for subsequent rounds.

Our approach. Building on these developments, we propose closing the agent, feedback, and environment loop. By combining model-level continual learning through agent–environment interaction with environment-level evolution, we argue that agents should continually learn at the frontier of dynamically evolving environments.

3 PRELIMINARY

Environment. The coding agent's environment is a self-contained, isolated, executable sandbox. For a given task $x \in \mathcal{X}$, the sandbox environment $\mathcal{E}$ can be represented as follows:

$$\mathcal{E}_x = (I_x, M_x, W_x, V_x), \tag{1}$$

where I_x provides the task instruction (`instruction.md`); M_x specifies the sandbox configuration (`task.toml`); W_x defines the runtime environment and initial codebase (`Dockerfile`); and V_x defines the private verification procedure (`test.sh` and `test_state.py`).

To support environment evolution, we extend existing environment to a dynamic version indexed by the training round t, allowing both the task instructions $x^{(t)}$ and the codebase $W_x^{(t)}$ to evolve across rounds. We introduce a difficulty knob d and a scaffolding knob s to govern the evolution. The difficulty knob mainly adjusts x and W_x with varying intent ambiguity, persona, conditions, and milestones. The scaffolding knob rewrites x and W_x while preserving the objective and acceptance criteria of the current task. These knobs are updated across training rounds by the curriculum operator $\mathcal{O}$ defined below.

LLM agents. We model an LLM agent as a policy $\pi_\theta(a_i \mid h_i)$ that interacts with an environment over T turns to generate an action sequence $a = [a_1, \ldots, a_T]$. Here, h_i denotes the context assembled by the harness and available within the agent's context window at turn i, which may include tool schema, relevant files, execution and terminal logs, and retained or summarized interaction history. A trajectory τ is one complete task attempt, including model messages, tool calls, and environment responses. A group $\mathcal{G}_\pi(x)$ is a collection of independent trajectories generated for the same prompt for group-based post-training.

Task states. Let $q_t(x)$ denote the pass@k score of policy π_t on task x with a group of rollouts $\mathcal{G}_\pi(x)$. For thresholds $0 \le \alpha < \beta \le 1$, we define task state z_t as:

$$z_t(x) = \begin{cases} \text{U (Unsolved)}, & q_t(x) \le \alpha, \\ \text{F (Frontier)}, & \alpha < q_t(x) < \beta, \\ \text{S (Saturated)}, & q_t(x) \ge \beta. \end{cases} \tag{2}$$

Task states are therefore relative to the current policy and a fixed evaluation configuration.

4 METHOD

At its core, FRONTIER-RSI enables agents to continually evolve at their *capability frontier*. Our closed loop consists of agent continual learning (Section 4.1) → task states (Section 3) → environment evolution (Section 4.2).

4.1 AGENT CONTINUAL LEARNING

Overview. We follow the existing agentic RL mechanism (Wang et al., 2026d;b) for agent training. Given the task x, we first collect independent trajectories using the rollout policy: $\mathcal{G}_\pi(x) = \{\tau_i\}_{i=1}^G$, the environment then runs tests on the final codebase to produce raw rewards $\{r_i\}_{i=1}^G$, which are typically binary. We perform credit assignment by shaping rewards according to rollout outcomes, execution status, and the agent's intermediate behavior at each interaction turn. Then, we train the policy model with GSPO (Zheng et al., 2025) and dynamic sampling following DAPO (Yu et al., 2025), filtering out samples with uniform raw rewards.

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{X},\, \{\tau_i\}_{i=1}^G \sim \pi_{\text{old}}(\cdot|\mathcal{E}_x(d,s))} \left[\frac{1}{G} \sum_{i=1}^{G} \min\{\rho_i(\theta)\, A_i,\ \text{clip}(\rho_i(\theta),\, 1-\epsilon_{\text{low}},\, 1+\epsilon_{\text{high}})\, A_i\} \right], \tag{3}$$

where A_i is the group-relative advantage of trajectory τ_i, and $\rho_i(\theta) = \left(\frac{\pi_\theta(\tau_i|\mathcal{E}_x(d,s))}{\pi_{\text{old}}(\tau_i|\mathcal{E}_x(d,s))} \right)^{\frac{1}{|\tau_i|}}$. The importance ratio, the clipping, and the optimization are therefore all applied to a whole trajectory, which is also the unit at which the verifier assigns reward. The length exponent keeps ρ_i on a comparable scale across episodes whose action sequences differ by orders of magnitude.

Policy Stability. Both the train–test mismatch in the open-source framework we use and the adoption of asynchronous RL to accelerate training can destabilize policy updates. We measure this gap where it arises and let it modulate the update. Let $\mathcal{T}_i$ be the action tokens of τ_i, $N_i = \max(1, |\mathcal{T}_i|)$, and $a_{i,t}, h_{i,t}$ the token at position $t \in \mathcal{T}_i$ and the history it was conditioned on.

$$\rho_i(\theta) = \exp\left[\frac{1}{N_i}\sum_{t\in\mathcal{T}_i}\log\frac{\pi_\theta(a_{i,t} \mid h_{i,t})}{\pi_{\text{old}}(a_{i,t} \mid h_{i,t})}\right], \tag{4}$$

with $d_{i,t}$, the per-position log-discrepancy between sampler and learner,

$$\begin{aligned} d_{i,t} &= \log \pi_{\text{old}}(a_{i,t} \mid h_{i,t}) - \log \pi_{\text{rollout}}(a_{i,t} \mid h_{i,t}), \\ w_{i,t} &= \text{stopgrad}\Big[v_i\, e^{\text{clip}(d_{i,t}, d_{\min}, d_{\max})}\mathbf{1}\{d^-_{\text{acc}} \le d_{i,t} \le d^+_{\text{acc}}\}\Big]. \end{aligned} \tag{5}$$

The exponential of $d_{i,t}$ is a truncated importance correction: clipping to $[d_{\min}, d_{\max}]$ keeps one bad position from dominating a trajectory, and the indicator drops positions outside $[d^-_{\text{acc}}, d^+_{\text{acc}}]$ instead of reweighting them, since a large correction is where the correction is least reliable. The veto v_i discards a trajectory outright once too much of $\mathcal{T}_i$ is severely mismatched. Trajectory i then enters Equation 3 scaled by ω_i, the mean of $w_{i,t}$ over surviving positions and 0 if none survive; $w_{i,t}$ is detached, so the optimizer cannot act on the weighting rule itself.

Context Compression. Long agentic episodes outrun the context window, especially for earlier foundation models, so we compact the policy-visible history with a deterministic *U-shaped* rule: the first and last few turns are kept verbatim, and the middle is replaced by compact tool-call skeletons with truncated string arguments. Compaction rewrites only what the policy reads; the persistent terminal and file-system state is untouched, so dropping an observation from the context never undoes the execution that produced it.

Three choices keep this from destabilizing the update. First, a candidate is committed only when its saving is measured on the re-rendered chat input rather than estimated from the removed turns, and only after a cooldown has elapsed or when the next completion would otherwise not fit. Second, replay aligns the stored sampled tokens against the final rendering and masks everything whose provenance cannot be established, namely tool observations together with rewritten, inserted, and diverged spans, so that a compaction artifact can never become a training target. Third, a compacted rollout still receives a single outcome advantage A_i over its eligible action tokens rather than one advantage per inter-compaction segment, so the number of compaction events does not scale a trajectory's gradient contribution. Thresholds and the exact eligibility rule are in Appendices D and E.

4.2 Environment Evolution.

Curriculum operator. At round t, we train the policy π_t on the task set $\mathcal{X}^{(t)}$ to obtain π_{t+1}, then evolve the environments according to their task states z_t using a curriculum operator $\mathcal{O}$. For tasks that are too difficult, it adjusts the scaffolding knob, transforming *unsolved* tasks into *prefix* tasks. For tasks that are too easy, $\mathcal{O}$ increases the difficulty knob, transforming *saturated* tasks into *escalation* tasks. These updates aim to place as many tasks in $\mathcal{X}^{(t+1)}$ as possible within the updated policy's learnable frontier. For an individual task $x^{(t)} \in \mathcal{X}^{(t)}$, the evolution is expressed as:

$$\mathcal{E}_{x^{(t+1)}}(d_{t+1}, s_{t+1}) = \mathcal{O}\Big(\mathcal{E}_{x^{(t)}}(d_t, s_t),\, z_t(x^{(t)})\Big), \tag{6}$$

Unsolved → Prefix Tasks. We aim to make unsolved tasks learnable by reducing their difficulty. Given a replayable successful trajectory $\tau^\star$ with ordered milestones $(m_1, \ldots, m_L)$ constructed by powerful models (*e.g.,* Fable 5), the prefix task $x_{\le j}$ requires completion of only the last $L - j$ milestones, where $1 \le j < L$. Thus, j is a candidate cut point for scaffolding. The resulting filesystem is materialized as a new initial environment image, implementing the scaffolding knob $s \to s^{(j)}$. The levels of the scaffolding knob naturally align with the task milestones. The agent receives the original instruction and continues from the saved intermediate state. The reference solution is not included in the prompt. We record the parent task, source trajectory, truncation point, and image identity to make each transformation traceable.

The instruction and verifier are retained, so the supplied progress reduces the remaining execution burden while preserving the full completion criterion. The reference trajectory remains private. Each

prefix environment must remain both unfinished and solvable. We validate the registered environment by requiring an initial verifier reward below 0.99, followed by the successful execution of a trusted recovery procedure with exit code zero and a final verifier reward of 0.99. In particular, we reject snapshots that already satisfy the verifier at initialization: such states leave no substantive task for the agent to complete and can produce misleading failure observations when the agent simply fails to submit. The details of verification are in Appendix B, and please refer to Figure 6 for demonstration.

Saturated → Escalation Tasks. For too easy tasks, we need to increase their difficulty. We distinguish between two cases. The first involves our internal long-horizon coding tasks, which typically span 3–5 pull requests (PRs). For these tasks, simply reducing the reasoning hints provided along the way naturally increases the difficulty. Here, we focus primarily on the second case: how to implement the difficulty knob $d \to d^+$ to increase the difficulty of tasks from open-source coding datasets through meaningful adversarial attacks. We construct harder variants by introducing recoverable obstacles into the environment, strengthening the public task requirements, or combining both transformations.

First, we construct state-based variants using three types of deterministic perturbations: (i) *failed-prefix perturbation*, which creates a recoverable but incomplete intermediate state; (ii) *hunk regression*, which reverts selected implementation fragments and requires the agent to locate and repair the resulting regressions; and (iii) *partial migration*, which leaves the system in a partially migrated state and requires the agent to complete the remaining migration steps. Each perturbation is applied to the parent environment and saved as a new environment image. These variants retain the parent verifier and typically preserve the original task instruction, increasing difficulty through changes to the starting state.

Second, we construct variants by adding explicit, more demanding constraints to the public task specification while retaining the original environment image. Requirement changes can be combined with state perturbations to produce compound variants. The parent verifier is retained when it remains applicable; changes that introduce new success criteria require a corresponding verifier update through the joint task-rewriting pipeline. This keeps the public requirements and evaluation criteria aligned.

Every candidate must pass static checks and validation in an instantiated environment. We require that the initial state fails verification while a trusted recovery procedure executes successfully and produces a verifier-passing state. These checks establish that the candidate is initially unsolved but remains solvable. We then evaluate the candidate using a frozen policy checkpoint, excluding infrastructure failures from the count. The details of verification can be seen in Appendix B, and please refer to Figure 7 for demonstration.

Frontier → Rewritten Tasks. We aim to keep them but refresh them. Starting from a parent task, we jointly rewrite the public instruction, reference solution, and verifier, together with any necessary environment fixtures. This produces new challenges grounded in the existing workspace while maintaining consistency between the requested behavior and a valid solution.

Let $\mathrm{Cand}_t(x)$ contain the validated variants of parent x. We still use the successful rate for calibration, estimated from the group $\mathcal{G}_{\pi_{t+1}}(x')$:

$$\widehat{q}_{t+1}(x') = \frac{\sum_{\tau \in \mathcal{G}_{\pi_{t+1}}(x')} C_{x'}(\tau)}{\left|\mathcal{G}_{\pi_{t+1}}(x')\right|}. \tag{7}$$

Infrastructure-invalid attempts are excluded. The children admitted for training are

$$\mathcal{C}_t(x) = \left\{ x' \in \mathrm{Cand}_t(x) : \alpha' < \widehat{q}_{t+1}(x') < \beta' \right\}. \tag{8}$$

For the current pool $\mathcal{X}_t$, let $\mathcal{P}_t = \{x \in \mathcal{X}_t : \mathcal{C}_t(x) \neq \varnothing\}$. We construct the next pool as

$$\mathcal{X}_{t+1} = (\mathcal{X}_t \setminus \mathcal{P}_t) \cup \bigcup_{x \in \mathcal{P}_t} \mathcal{C}_t(x). \tag{9}$$

Retaining parents without accepted replacements preserves training coverage when adaptation is unsuccessful. Sampling assigns target mass λ to each evolved parent and unit mass to each retained parent, with an evolved parent's mass divided among its accepted children. Every accepted child is included at least once per epoch. Our experiments use calibration groups of size $K = 16$, $(\alpha, \beta) = (0.25, 0.75)$, and $\lambda = 4$. Evaluation environments remain fixed when measuring task-state flows. More details of verification can be seen in Appendix B.

Method	Terminal-Bench 2			TerminalWorld			SWE-Bench Verified	SWE-Bench Pro†	Mean
	Pass@1	Pass@4	Pass@8	Pass@1	Pass@4	Pass@8			
`Qwen3-8B`	3.2	7.9	10.1	13.0	24.5	31.0	6.2	2.0	12.2
+ Round 1	4.4	10.6	14.6	16.0	30.3	38.0	7.6	4.0	15.7
Harness-level RSI									
Meta-Harness	4.1 ↓6.8%	9.5 ↓10.4%	13.5 ↓7.5%	15.4 ↓3.7%	28.1 ↓7.3%	36.0 ↓5.3%	6.6 ↓13.2%	2.0 ↓50.0%	14.4 ↓8.3%
HarnessBridge	4.2 ↓4.5%	10.0 ↓5.7%	14.6 –0%	15.2 ↓5.0%	28.4 ↓6.3%	36.5 ↓3.9%	6.6 ↓13.2%	2.0 ↓50.0%	14.7 ↓6.4%
Model-level RSI									
SSR									
+ Round 2	4.5 ↑2.3%	11.2 ↑5.7%	16.9 ↑15.8%	18.2 ↑13.7%	33.4 ↑10.2%	42.0 ↑10.5%	8.2 ↑7.9%	4.0 –0%	17.3 ↑10.2%
+ Round 3	4.6 ↑4.5%	11.7 ↑10.4%	18.0 ↑23.3%	18.4 ↑15.0%	33.7 ↑11.2%	42.5 ↑11.8%	8.6 ↑13.2%	5.0 ↑25.0%	17.8 ↑13.4%
Environment-level RSI									
RST									
+ Round 2	4.4 –0%	10.6 –0%	15.7 ↑7.5%	16.0 –0%	30.3 –0%	39.0 ↑2.6%	7.8 ↑2.6%	6.0 ↑50.0%	16.2 ↑3.2%
+ Round 3	4.4 –0%	10.6 –0%	16.9 ↑15.8%	17.4 ↑8.7%	32.0 ↑5.6%	40.5 ↑6.6%	8.0 ↑5.3%	7.0 ↑75.0%	17.1 ↑8.9%
Ours									
+ Round 2	4.5 ↑2.3%	11.9 ↑12.3%	18.0 ↑23.3%	21.6 ↑35.0%	37.1 ↑22.4%	43.0 ↑13.2%	8.6 ↑13.2%	6.0 ↑50.0%	18.8 ↑19.7%
+ **Round 3**	**5.6** ↑27.3%	**13.8** ↑30.2%	**22.5** ↑54.1%	**22.8** ↑42.5%	**37.9** ↑25.1%	**45.0** ↑18.4%	**10.2** ↑34.2%	**8.0** ↑100.0%	**20.7** ↑31.8%

Table 1: Overview of RSI on `Qwen3-8B` trained with TMax. The shaded headers mark the task formats which match TMax. Relative change is against the + Round 1 row.

5 Experiment

5.1 Experiment Setup

Seed Task Dataset. We initialize the first round of coding-agent training with seed tasks, keeping both the difficulty and scaffolding knobs at their initial settings. We consider two seed datasets: TMax (Ivison et al., 2026) is a public dataset for training terminal agents, from which we use only 5.8% of the tasks, and Cross-PR is our internal dataset constructed from multiple pull requests within a single codebase. By spanning changes across multiple pull requests, Cross-PR tasks involve more complex construction and longer solution horizons than conventional coding tasks. More details are shown in Appendix A.

Evaluation Benchmarks SWE-Bench Verified (Jimenez et al., 2024; OpenAI, 2024) comprises 500 human-validated repository-level issues and measures in-domain generalization to repositories and issues. SWE-Bench Pro (Deng et al., 2025) evaluates more challenging, long-horizon software engineering tasks involving changes across multiple files. It is noted that we only evaluate throughout on a high-quality subset of 100 tasks. Terminal-Bench (Merrill et al., 2026) measures OOD transfer through diverse command-line tasks requiring sustained interaction with stateful environments, with success verified by tests of the final environment state. TerminalWorld (Chu et al., 2026) reverse-engineers the same terminal setting from in-the-wild terminal recordings, yielding 1,530 validated tasks over 18 real-world categories and 1,280 distinct commands. We report results on the manually reviewed verified subset of 200 representative tasks.

Baselines. To comprehensively evaluate FRONTIER-RSI, we select representative existing methods with open-source implementations as baselines. Specifically, we include META-HARNESS and HARNESSBRIDGE at the harness level, SSR at the model level, and RST at the environment level.

Implementation Details. To thoroughly assess the effectiveness of our evolution loop, we conduct experiments varying both the difficulty of the seed tasks used to initialize training and the agent's foundation model. Specifically, we utilize `Qwen3-8B` (Qwen Team, 2025) and `Qwen3.6-35B-A3B` (Qwen Team, 2026) MoE models, training each on datasets of varying difficulty. We use four GPU nodes, each equipped with eight NVIDIA H200 GPUs, allocating three nodes to rollout generation and one to policy training. All sandboxes are deployed on separate Amazon EC2 CPU nodes, decoupling environment execution from GPU-based training. We use the SLIME (Zhu et al., 2025) framework for reinforcement learning. Before RL training, we warm-start the policy using high-quality rollout trajectories obtained through self-distillation. Full implementation details are provided in Appendices D and E. More details on evaluation are in Appendix F

5.2 Main Results

Table 1 evaluates `Qwen3-8B` trained with TMax against the three families of RSI baselines. Harness-level baselines are implemented on the base model. Three observations are below. **(1)** Editing the harness is no substitute for training. Both harness-level methods improve the base policy, and both

Method	Terminal-Bench 2			TerminalWorld			SWE-Bench Verified	SWE-Bench Pro†	Mean
	Pass@1	Pass@4	Pass@8	Pass@1	Pass@4	Pass@8			
`Qwen3.6-35B-A3B`	34.5	49.6	52.2	43.5	65.0	72.5	61.8	39.0	52.3
+ Round 1	37.1	51.8	59.6	50.0	72.0	79.0	68.8	45.0	57.9
Harness-level RSI									
Meta-Harness	36.4 ↓1.9%	49.6 ↓4.2%	55.1 ↓7.6%	45.5 ↓9.0%	67.9 ↓5.7%	76.5 ↓3.2%	62.6 ↓9.0%	40.0 ↓11.1%	54.2 ↓6.4%
HarnessBridge	36.5 ↓1.6%	49.6 ↓4.2%	55.1 ↓7.6%	45.7 ↓8.6%	68.7 ↓4.6%	76.5 ↓3.2%	63.0 ↓8.4%	43.0 ↓4.4%	54.8 ↓5.4%
Model-level RSI									
SSR									
+ Round 2	37.6 ↑1.3%	54.1 ↑4.4%	64.0 ↑7.4%	52.5 ↑5.0%	74.5 ↑3.5%	81.5 ↑3.2%	69.4 ↑0.9%	46.0 ↑2.2%	60.0 ↑3.6%
+ Round 3	38.1 ↑2.7%	56.7 ↑9.5%	67.4 ↑13.1%	52.1 ↑4.2%	74.8 ↑3.9%	82.0 ↑3.8%	69.4 ↑0.9%	45.0 –0%	60.7 ↑4.8%
Environment-level RSI									
RST									
+ Round 2	36.9 ↓0.5%	51.3 ↓1.0%	58.4 ↓2.0%	52.8 ↑5.6%	75.2 ↑4.4%	82.5 ↑4.4%	69.0 ↑0.3%	46.0 ↑2.2%	59.0 ↑1.9%
+ Round 3	37.2 ↑0.3%	52.4 ↑1.2%	60.7 ↑1.8%	53.9 ↑7.8%	76.7 ↑6.5%	84.0 ↑6.3%	69.2 ↑0.6%	46.0 ↑2.2%	60.0 ↑3.6%
Ours									
+ Round 2	38.1 ↑2.7%	55.8 ↑7.7%	67.4 ↑13.1%	54.5 ↑9.0%	78.1 ↑8.5%	85.5 ↑8.2%	69.6 ↑1.2%	47.0 ↑4.4%	62.0 ↑7.1%
+ **Round 3**	**38.3** ↑3.2%	**57.0** ↑10.0%	**69.7** ↑16.9%	**55.1** ↑10.2%	**78.6** ↑9.2%	**86.0** ↑8.9%	**70.0** ↑1.7%	**48.0** ↑6.7%	**62.8** ↑8.4%

Table 2: Ablation on long-horizon coding training. Relative change is against the + Round 1 row.

Method	Terminal-Bench 2			TerminalWorld			SWE-Bench Verified	SWE-Bench Pro†	Mean
	Pass@1	Pass@4	Pass@8	Pass@1	Pass@4	Pass@8			
Ours w/o Evolution									
+ Round 2	4.1	9.5	13.5	15.5	28.3	34.0	7.2	5.0	14.6
+ Round 3	3.9	8.9	12.4	15.2	27.3	32.0	3.4	2.0	13.1
Ours									
+ Round 2	4.5	11.9	18.0	21.6	37.1	43.0	8.6	6.0	18.8
+ **Round 3**	**5.6**	**13.8**	**22.5**	**22.8**	**37.9**	**45.0**	**10.2**	**8.0**	**20.7**

Table 3: Ablation on environment evolution, on `Qwen3-8B` trained with TMax.

stay below the 15.7 of a single training round. With the weights frozen, a harness can only reorganize competence the policy already has. **(2)** Open-loop environment expansion underperforms model-level RSI. RST trails SSR at both matched rounds, 16.2 against 17.3 and 17.1 against 17.8, though it is the method that grows the task pool. Generating executable tasks is not the same as generating tasks the current policy can learn from: the expansion is open-loop, so a task may land where the policy already succeeds or where it still fails outright, and either way the group has uniform outcomes and is discarded by the dynamic filter. **(3)** FRONTIER-RSI leads every baseline on every benchmark. After three rounds FRONTIER-RSI reaches 20.7 in mean, 2.9 points above the strongest baseline (+16.3%) and 5.0 above the shared first round. The margin holds on every benchmark: Terminal-Bench 2 Pass@8 22.5 against 18.0, TerminalWorld Pass@1 22.8 against 18.4, SWE-Bench Verified 10.2 against 8.6, SWE-Bench Pro 8.0 against 7.0.

5.3 ABLATION STUDY

Ablation on long-horizon coding task. Table 2 repeats the comparison on `Qwen3.6-35B-A3B` seeded with Cross-PR, a stronger base policy on longer-horizon tasks. The loop still evolves productively there: FRONTIER-RSI reaches 62.8 in mean, 4.9 points above the shared first round and 2.1 above the strongest baseline. The other conclusions carry over unchanged: both harness-level methods (54.2 and 54.8) stay below the 57.9 of a single training round, and RST again trails SSR at both matched rounds. FRONTIER-RSI gains 8.4% here against 31.8% on `Qwen3-8B`, and its margin over the best baseline narrows from 16.3% to 3.5%.

Ablation on key modules. Table 3 removes the curriculum operators. Without evolution the policy does not merely stop improving, it degrades. Mean falls to 14.6 after the second round and 13.1 after the third, below the 15.7 the shared first round had already reached. This comparison supports the benefit of adapting the training distribution over successive rounds. The curriculum operator broadens intervention to low- and high-success-rate tasks, including those that can still produce non-uniform reward groups, with the aim of providing more suitable subsequent learning challenges. The observed degradation on the fixed pool motivates this adaptation, although identifying its precise optimization mechanism requires further analysis.

5.4 ANALYSIS

Besides public benchmarks, we also present the validation set to validate the closed loop. We hold out 200 TMax tasks that are excluded from the training pool and from every evolution operator, drawn

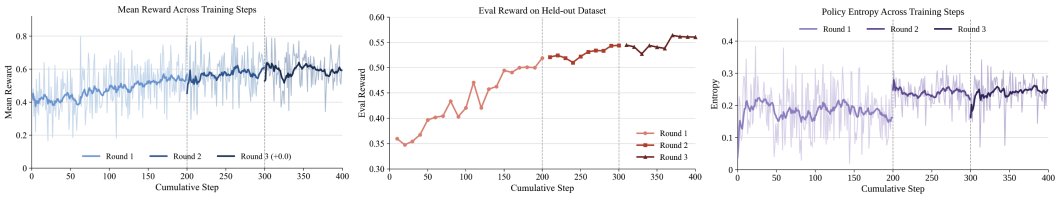

Figure 3: Training dynamics of FRONTIER-RSI on `Qwen3-8B` with TMax. Dashed lines mark the round boundaries, where the environment is evolved against the updated policy. **Left:** mean training reward. **Middle:** reward on the held-out set of 200 TMax tasks that never enter the training pool. **Right:** policy entropy. Faint traces are per-step values, solid traces their running mean.

uniformly across the corpus's different complexity levels so that the held-out set is not concentrated at the easy end. Our analysis is based on the results on the held-out dataset.

The loop improves the policy with diminishing marginal returns. The middle panel of Figure 3 reports the reward on the validation set. It rises from 0.35 at initialization to 0.52 by the end of the first round and to 0.56 after the third, and it rises monotonically across the round boundaries rather than only within a round. The gains therefore transfer off the evolved pool: the policy is learning the competence the tasks are built to elicit, not the particular instances it was trained on. The per-round increments do shrink, $+0.17$, $+0.025$ and $+0.015$, which is the behavior the frontier signal predicts, since each round starts from a policy for which fewer of the remaining seed tasks are still unsolved. However, we also observe diminishing gains across successive rounds, consistent with the trends on public benchmarks. This suggests that evolving a fixed pool of data has a ceiling and cannot sustain unbounded improvement, echoing the diminishing returns associated with data scaling laws.

The learnable frontier is stable while it advances. The held-out tasks in Figure 4 are fixed and never evolved, so a task changes state only because the policy changed, and the partition measured after each round is a direct readout of where that frontier sits. It moves in the intended direction throughout. Unsolved groups fall from 48% before training to 31% after the third round, while saturated groups rise from 20% to 31%. Meanwhile, the fraction of tasks in the frontier band remains between 32% and 38% across the evaluated rounds. Together with the decrease in low-success-rate tasks and the increase in high-success-rate tasks, this pattern is consistent with improving capability on the fixed held-out set while a substantial fraction of tasks remains at intermediate success rates. These proportions summarize the policy-relative task distribution; the fraction of rollout groups retained for training is determined separately by their observed reward variation.

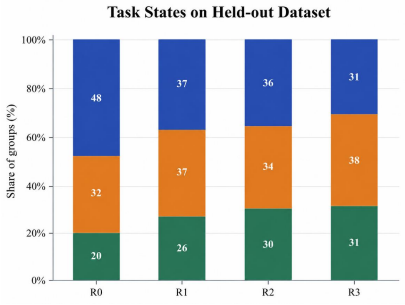

Figure 4: Task states on the held-out set after each round.

Evolved tasks restore exploration. The right panel tracks policy entropy. During the first round entropy decays from about 0.20 to 0.17 as the policy commits to the strategies that work on a fixed pool, the usual precursor to a collapsed, near-deterministic policy that stops producing new behavior to learn from. At the first re-evolution entropy jumps to about 0.26 and then holds near 0.24 through the third round, without an entropy bonus large enough to force it there. The rebuilt tasks are unfamiliar enough that the policy has to explore again, so the environment side of the loop supplies the exploration pressure that a static pool bleeds away.

6 CONCLUSION

We presented FRONTIER-RSI, a framework in which the environment is evolved against a measurement of the learner rather than independently of it. After each round the updated policy relabels every task as unsolved, frontier or saturated, and a curriculum operator moves the pool back toward the learnable band, escalating what has saturated and supplying verified intermediate progress for what remains out of reach. Extensive experiments on various coding tasks have verified our method.

USE OF LARGE LANGUAGE MODELS

We used an LLM-based coding assistant throughout the project to refine the code and revise the manuscript. We disclose this assistance in accordance with ICLR's policy. No LLMs were used to fabricate citations or supporting evidence.

ETHICS STATEMENT

This work is intended exclusively for academic research, and use for other purposes is prohibited. Users bear sole responsibility for the content generated by the models. The current implementation does not include safeguards against harmful, unfaithful, biased, or toxic outputs. Modified models must carry distinct version numbers to ensure traceability to the originals described in this manuscript. The authors accept no responsibility for consequences arising from model use.

REPRODUCIBILITY STATEMENT

The project code is available at https://anonymous.4open.science/r/Frontier-RSI-1C0D/. The public datasets used in this work are openly available. All foundation models we use are open-source and readily accessible.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009.

Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in ai: From bounded self-refinement to autonomous research loops. *arXiv preprint arXiv:2607.07663*, 2026a.

Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in AI: From bounded self-refinement to autonomous research loops. *arXiv preprint arXiv:2607.07663*, 2026b. URL https://arxiv.org/abs/2607.07663.

Zhaoyang Chu, Jiarui Hu, Xingyu Jiang, Pengyu Zou, Han Li, Chao Peng, Peter O'Hearn, Earl T. Barr, Mark Harman, Federica Sarro, and He Ye. Terminalworld: Benchmarking agents on real-world terminal tasks, 2026. URL https://arxiv.org/abs/2605.22535.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks?, 2025. URL https://arxiv.org/abs/2509.16941.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, 2020.

Alex Graves, Marc G Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *International Conference on Machine Learning*, 2017.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. *arXiv preprint arXiv:2508.05004*, 2025. URL https://arxiv.org/abs/2508.05004.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026. URL https://arxiv.org/abs/2606.23321.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, 2021.

Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. Socratic-Geo: Synthetic data generation and geometric reasoning via multi-agent interaction. *arXiv preprint arXiv:2602.03414*, 2026. URL https://arxiv.org/abs/2602.03414.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

kvcache-ai. AgentENV documentation: Overview. https://kvcache-ai.github.io/AgentENV/latest/getting-started/overview.html. Accessed: 2026-09-15.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026. URL https://arxiv.org/abs/2603.28052.

Dinging Li, Yingxiu Zhao, Xinrui Cheng, Kangheng Lin, Hongbo Peng, Hongxing Li, Zixuan Wang, Yuhong Dai, Haodong Li, Jia Wang, Yukang Shi, Liang Zhao, Jianjian Sun, Zheng Ge, Xiangyu Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SpatialEvo: Self-evolving spatial intelligence via deterministic geometric environments. *arXiv preprint arXiv:2604.14144*, 2026a. URL https://arxiv.org/abs/2604.14144.

Zhongzhi Li, Yucheng Shi, Zongxia Li, Ruhan Wang, Anhao Li, Zixun Huang, Junyao Yang, Lei Ke, Ninghao Liu, Haitao Mi, and Leowei Liang. Recursive synthesis for long-horizon terminal tasks. *arXiv preprint arXiv:2608.05466*, 2026b. URL https://arxiv.org/abs/2608.05466.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*, 2025. URL https://arxiv.org/abs/2506.24119.

Mike A. Merrill, Alex G. Shaw, Nicholas Carlini, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2601.11868.

OpenAI. Introducing SWE-bench verified. https://openai.com/index/introducing-swe-bench-verified/, 2024. Accessed: 2026-08-21.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, 2022.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Qijia Shen, Zhiqi Huang, Vamsidhar Kamanuru, Aznaur Aliev, Jay Rainton, Ahmed Awelkair, Zhichen Zeng, Jiajun Li, Shi Dong, Yueming Yuan, Boyuan Ma, Qizheng Zhang, Jiwei Fu, Yuzhen Mao, Wendong Fan, Ping Nie, Philip Torr, Bernard Ghanem, Changran Hu, Jonathan Lingjie Li, Urmish Thakker, and Guohao Li. SETA: Scaling environments for terminal agents. *arXiv preprint arXiv:2607.10891*, 2026. URL https://arxiv.org/abs/2607.10891.

Haixin Wang, Hejie Cui, Chenwei Zhang, Xin Liu, Shuowei Jin, Shijie Geng, Xinyang Zhang, Nasser Zalmout, Zhenyu Shi, and Yizhou Sun. T^2PO: Uncertainty-guided exploration control for stable multi-turn agentic reinforcement learning. *arXiv preprint arXiv:2605.02178*, 2026a.

Haixin Wang, Xiaoxuan Wang, Junkai Zhang, Han Zhang, Renliang Sun, Alexander K Taylor, Yidan Shi, Haoran Deng, Chenguang Wang, Jason Cong, Yizhou Sun, and Wei Wang. Aclarena: Agent continue learning in multi-stage post-training, 2026b. URL https://arxiv.org/abs/2609.23989.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Shaobo Wang, Zhengbo Jiao, Zifan Zhang, Yilang Peng, Xu Ze, Boyu Yang, Wei Wang, Hu Wei, and Linfeng Zhang. Socratic-Zero: Bootstrapping reasoning via data-free agent co-evolution. *arXiv preprint arXiv:2509.24726*, 2025. URL https://arxiv.org/abs/2509.24726.

Xiaoxuan Wang, Haixin Wang, Alexander Taylor, Jason Cong, Yizhou Sun, and Wei Wang. Harness-bridge: Learnable bidirectional controller for llm agent harness. *arXiv preprint arXiv:2606.12882*, 2026c. URL https://arxiv.org/abs/2606.12882.

Xiaoxuan Wang, Han Zhang, Haixin Wang, Yidan Shi, Ruoyan Li, Kaiqiao Han, Chenyi Tong, Haoran Deng, Renliang Sun, Alexander Taylor, et al. Arlarena: A unified framework for stable agentic reinforcement learning. *arXiv preprint arXiv:2602.21534*, 2026d.

Yuxiang Wei, Zhiqing Sun, Emily McMilin, Jonas Gehring, David Zhang, Gabriel Synnaeve, Daniel Fried, Lingming Zhang, and Sida Wang. Toward training superintelligent software agents through self-play swe-rl. *arXiv preprint arXiv:2512.18552*, 2025.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Advances in Neural Information Processing Systems*, 38:78500–78525, 2026.

Chuan Xiao, Zhengbo Jiao, Shaobo Wang, Wei Wang, Bing Zhao, Hu Wei, Linfeng Zhang, and Lin Qu. Socratic-SWE: Self-evolving coding agents via trace-derived agent skills. *arXiv preprint arXiv:2606.07412*, 2026. URL https://arxiv.org/abs/2606.07412.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 1(2), 2025.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute Zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025. URL https://arxiv.org/abs/2505.03335.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

Tong Zheng, Xidong Wu, Zheng Zhang, Zhankui He, Chaoyi Zhang, Benjamin Coleman, Ruoqiao Wei, Di Bai, Haolin Liu, Rui Liu, et al. Dream-rsi: Recursive self-improvement through evolving worlds. *arXiv preprint arXiv:2609.14858*, 2026.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.