VERA: SCALING VERIFIABLE ENVIRONMENTS FOR AGENTIC CO-EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Competent agents need precise, verifiable environments: sandboxes that are resumable at any stage and evolve from observable evidence. Long-horizon work exposes how rare these are: a medical research agent must ground a finding, classify it, and write a report over dozens of dependent steps, yet recent environments score only the outcome. We present VERA, which builds such environments at scale and lets agents evolve on them. VERA builds these environments from initial trajectories: an agent writes rubrics, executable checks, a judge verifies each sandbox, and only those that pass enter the training bank. On these environments, VERA alternates between two updates: train the model with rubric rewards, or edit the harness skills; a verifier accepts each change only if it improves. This attribution distinguishes VERA's co-evolution: updates target the cause, not just the outcome. We open-source 9,000+ long-horizon verifiable environments. At 9B, the co-evolved agent beats the strongest baseline by 10.3 and 13.0 points in the two domains. At 27B, it surpasses the baseline on AutoCoWorkBench (71.6) and AutoMedBench (80.7), transfers to unseen workflows, and retains general capabilities.

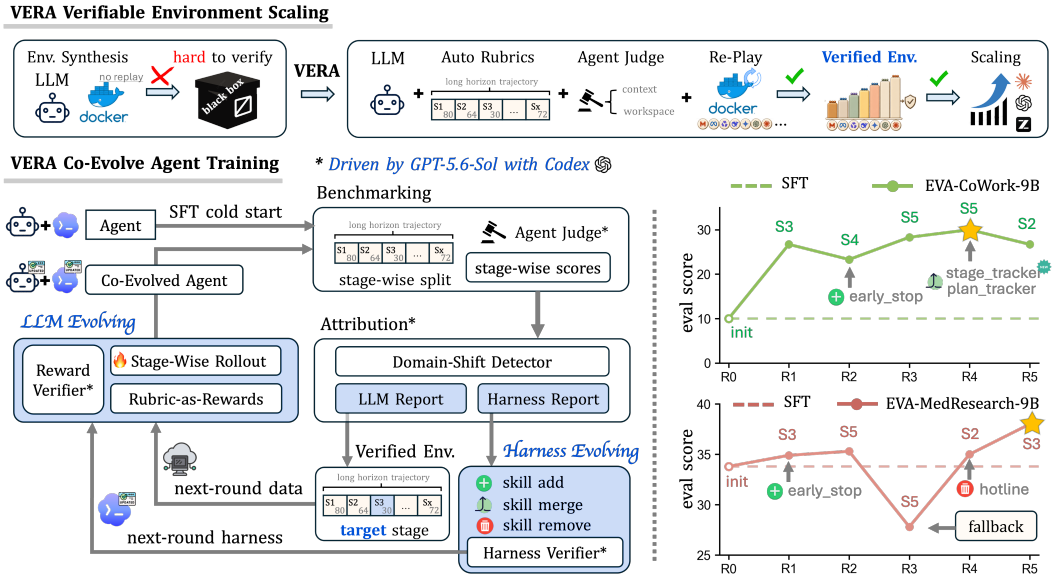


Figure 1: **VERA enables model–harness co-evolution through verifiable environments. Top:** Auto-generated rubrics, workspace inspection by an Agent Judge, and container replay verify environments before scaling. **Bottom left:** Stage-wise scores and execution traces feed a Domain-Shift Detector for failure attribution. The LLM Report selects a target stage for training with rubric rewards checked by the Reward Verifier. The Harness Report guides skill additions, merges, and removals tested by the Harness Verifier. Model updates freeze the harness; harness updates freeze the model. Accepted changes feed the next round. **Right:** CoWork and MedResearch scores across rounds. Dashed lines mark SFT baselines; stage labels indicate training targets, skill labels mark harness edits, and fallback restores an accepted model checkpoint.

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.

1 INTRODUCTION

One ultimate ambition of agentic artificial intelligence is a system that improves from its own attempts and grows into an expert-level generalist. AI has taken visible steps toward this ambition in software engineering (Jimenez et al., 2024; Merrill et al., 2026), computer-use, office automation (Yao et al., 2025; Zhou et al., 2024). Yet the ambition remains unrealized for the long-horizon research. Consider medical research: a competent research agent must ground findings in an image, classify them, and compose a report under expert scrutiny, a workflow of dozens of dependent steps in which one early mistake invalidates the entire workflow. Unlike coding and office workflows, where we can find trajectory resources online, very few AI systems today can learn to master research workflows because they lack practical environments. We argue that the bottleneck is not the learning algorithm but the *environment*.

For modern agent training, an environment is a reusable sandbox, a task packaged with its tools, workspace, and initial state, together with a mechanism that scores success from observable evidence. Reinforcement learning consumes such environments, and self-improvement is impossible without them. Current practice builds environments in two ways, and both break down on long-horizon agentic work. First, manual construction pipelines (Pan et al., 2024; Jain et al., 2025; Yang et al., 2025) synthesize executable tasks and define success by a test suite. Agents improve themselves through test signals, through RL with verifiable rewards (Shao et al., 2024; DeepSeek-AI, 2025), on-policy distillation (Agarwal et al., 2024), or reflection training (Bi et al., 2026); or through the harness, skills, tools, and execution rules, via verbal reflection (Shinn et al., 2023), skill libraries (Wang et al., 2023), prompt and workflow search (Khattab et al., 2024; Zhang et al., 2025b), and harness-code evolution (Hu et al., 2025; Zhang et al., 2025a; Lee et al., 2026). Recent systems co-evolve both (Wu et al., 2026b;a), but without verified evidence of a failure, each change is judged only by scores or pass rates. Second, evaluation benchmarks (Jimenez et al., 2024; Merrill et al., 2026; Chan et al., 2024; Schmidgall et al., 2024; Zuo et al., 2025) are also environments that are curated by hand and score only the outcome. This process does not measure rewards or localize failures in the intermediate stages. The result is a structural gap: environments yield no stage-level verified evidence, and self-evolution loops cannot gain from precision failures. In words, *no precise verifiable environments*.

In this paper, we present VERA (*Verifiable Environments for Agentic co-Evolution*), which closes this gap by constructing open verifiable environments and letting the agent evolve on them. Our starting observation is that benchmark trajectories already contain everything a verifiable environment starts with: initial states, tool interactions, and their observable effects. A data pipeline, AUTO-RUBRIC, converts such trajectories into training sandboxes. A coding agent writes binary rubrics over action arguments, execution effects, artifacts, and outputs; stage-wise sandboxes restore the workspace at the start of each stage: *plan*, *setup*, *verify*, *execute*, *submit*, so a long workflow decomposes into verifiable units, while end-to-end sandboxes use an LLM for tool calls (Zhang et al., 2026). Executable code checks and an agent judge inspect the workspace, and a failed executable check overrides any judge score. On these environments, the VERA loop evolves the agent in alternating rounds: *LLM Attribution* reads stage-wise rubric scores and execution traces to decide which step is failing; model rounds run full-parameter GRPO (Shao et al., 2024) with rubric rewards that a Reward Verifier checks against rollout evidence; harness rounds add, merge, or remove skills, accepted only when paired reruns with the frozen model confirm improvement. This design solves three problems: rewards come from what the agent actually did rather than from tests, long tasks are scored step by step, and only one part of the agent changes at a time, so every gain has clear evidence.

We evaluate one procedure in two independent domains, CoWork and Medical Research, with matched training budgets, a shared sandbox bank, and frozen benchmark scorers. At 9B, the co-evolved agent outperforms the strongest single-axis baseline by 10.3 Overall points on AutoCoWorkBench and 13.0 on AutoMedBench, and leads on SWE-Bench Verified and MedXpertQA Text under their official protocols; freezing either axis forfeits roughly half the gain. Scaling to Qwen3.8-27B, VERA reaches 71.6 Overall on AutoCoWorkBench, surpassing Claude Opus 4.8, and 80.7 on AutoMedBench, within 1.2 points of Claude Opus 4.8 with the highest Agentic score; it transfers out of domain to Automation-Bench (43.8%) and AgentClinic (76.4%), neither of which enters sandbox construction. Across 4B, 9B, and 27B backbones, domain gains grow with scale, and only at 27B do agents also retain, and mostly improve, general capabilities on AIME, ALFWorld, GPQA-Diamond, and

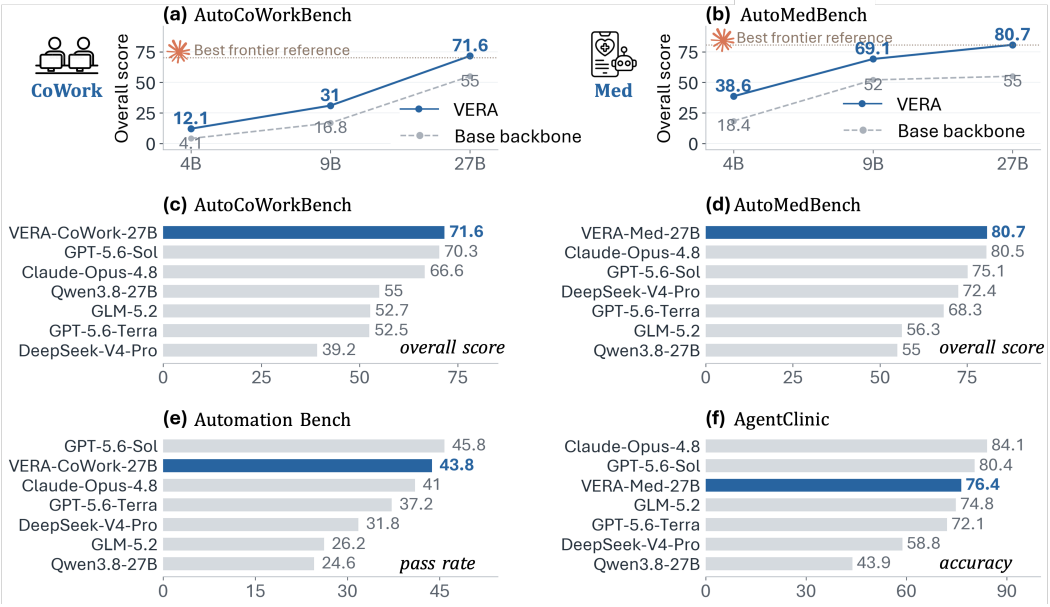


Figure 2: **VERA improves agents across model sizes and achieves competitive performance with a 27B backbone.** (a–b) Co-evolution improves on the base agents at 4B, 9B, and 27B; dotted lines mark the best frontier reference scores. (c–d) VERA-27B achieves the highest Overall scores among the compared agents on AutoCoWorkBench and AutoMedBench. (e–f) On Automation-Bench (public 600) (Zapier, 2026) and AgentClinic (Schmidgall et al., 2024), VERA-27B improves over its base agent by 19.2 and 32.5 percentage points, respectively, ranking second and third among the compared systems. These comparisons evaluate complete agents with their configured productivity harnesses, rather than model weights alone. Blue denotes VERA; gray denotes comparison models. Harness and reasoning-effort settings appear in Appendix D.1 (Table 13).

IF-Bench, indicating that strong backbones turn verified self-evolution into net capability rather than a trade-off against it.

To the best of our knowledge, we build the first large-scale verifiable environments for the joint self-evolution of an LLM and its harness, and we open-source the resulting 9,000+ environment corpus. To summarize, we make the following contributions:

- **VERA: verifiable environment scaling for recursive self-improvement.** We turn benchmark trajectories into restartable, rubric-scored sandboxes for long-horizon agent tasks, with an admission loop that accepts only environments executable and scoreable from observable evidence.
- **Closed-loop co-evolution of the LLM and its harness on VERA environments.** Rubrics provide verified evidence; LLM Attribution turns it into model and harness reports; the two components update in alternating rounds, and verifiers decide whether each update is kept.
- **An open corpus of 9,000+ verifiable long-horizon training environments and scaling to a 27B backbone.** At 9B the co-evolved agent exceeds the strongest single-axis baseline by 10.3 (CoWork) and 13.0 (MedResearch) Overall points; at 27B it surpasses Claude Opus 4.8 on AutoCoWorkBench and comes within 1.2 points on AutoMedBench.

2 METHOD

We introduce **VERA** (*Verifiable Environments for Agentic co-Evolution*). Its goal is to let a long-horizon agent improve itself from its own attempts, without human labels, through *rule-guided self-evolution*: verified rubrics provide the evidence, *LLM Attribution* turns the evidence into model-side and harness-side reports, and verifiers decide whether each update is kept. Concretely, VERA

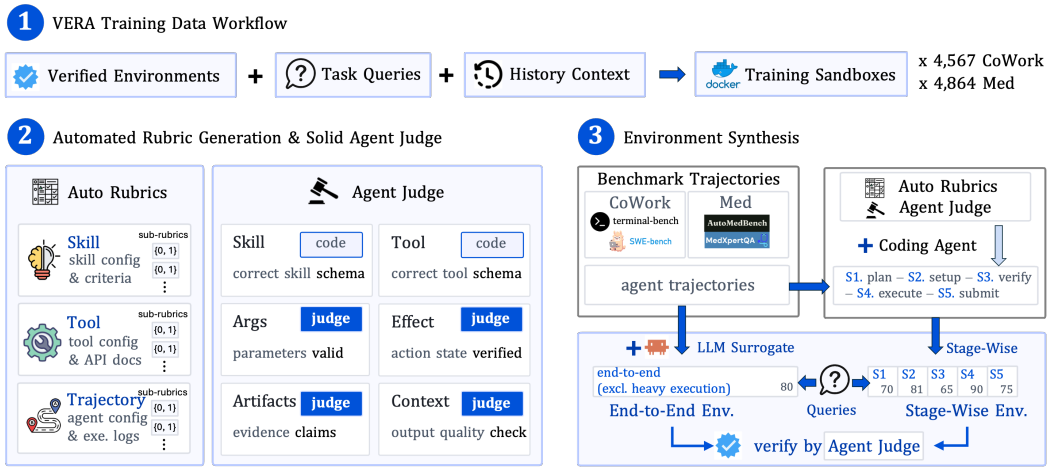


Figure 3: **AUTO-RUBRIC makes long workflows restartable and verifiable. (1) Training sandbox construction.** Each sandbox combines a verified environment (tools, workspace, and initial state), a task query, and the interaction history needed to resume the task. **(2) Rubric generation and evidence verification.** A coding agent generates binary rubric items from skill configurations, tool API documentation, and execution logs. Each item defines a success criterion scored in $\{0, 1\}$ from observable evidence. Executable checks validate skill and tool schemas. The Agent Judge inspects the trajectory and workspace to assess argument validity, actual execution effects, artifact evidence, and output quality in the task context. Failed executable checks override judge scores. **(3) Stage-wise and end-to-end environment synthesis.** Benchmark trajectories are segmented into five stages: plan, setup, verify, execute, and submit (S1–S5). Stage-wise environments restore each stage's starting state and attach the corresponding query, history, and rubrics, allowing that stage to be practiced and scored independently. End-to-end environments preserve the full workflow while an LLM surrogate simulates responses from time-consuming tool calls (Ren et al., 2025). Both types must pass executable checks and Agent Judge verification before entering the training bank. Numbers beside the environments illustrate rubric scores.

consists of two pipelines (Figure 1): a *data pipeline*, AUTO-RUBRIC, which turns benchmark trajectories into verified, restartable training environments (Figure 3); and a *training pipeline* that co-evolves the LLM and its harness on these environments in alternating rounds. The rest of this section presents each pipeline in turn.

2.1 VERA: CO-EVOLUTION METHOD

The VERA loop improves two parts of the agent in alternating rounds: the LLM and the harness (its skills, tools, and execution rules).

Model training freezes the harness; harness testing freezes the model; rubrics and verifiers stay fixed within a round. Any gain is therefore attributable to a single axis, which is what lets rubric evidence decide what to change next. To choose that axis, each round starts by evaluating the current pair on development tasks. *LLM Attribution* reads the stage-wise scores and execution traces and produces two reports: an *LLM Report* that locates the target stage, the stage with the most frequent model-attributed failures, and a *Harness Report* that lists missing or misleading procedures. Rubrics provide the evidence; verifiers decide acceptance. Infrastructure failures are excluded.

Update the model, freeze the harness. The model makes new attempts in the stage-wise and end-to-end sandboxes, and full-parameter GRPO (Shao et al., 2024) updates its weights with rubric scores as rewards, following the execution/training separation of Agent Lightning (Luo et al., 2025). The optimizer and budget match our GRPO baseline; only the learning signal differs: rewards come from stage rubrics, and the curriculum follows the target stage. For stage k, let $z_{k,j} \in \{0, 1\}$ be the score of rubric item j, $w_{k,j} > 0$ its weight, and $h_k \in \{0, 1\}$ indicate whether all mandatory checks

Table 1: **VERA jointly evolves the LLM and harness with rubric-based rewards and verification.** Weights and Harness indicate updated components. Reward and Verification indicate rubric-based training rewards and verified-rubric checks in our implementations.

Evolution	Method	Weights	Harness	Reward	Verification
Harness	ADAS (Hu et al., 2025)	✗	✓	✗	✗
	Ours (VERA Harness)	✗	✓	✗	✓
LLM	SFT (Ouyang et al., 2022)	✓	✗	✗	✗
	GRPO (Shao et al., 2024)	✓	✗	✓	✓
	RaR (Gunjal et al., 2025)	✓	✗	✓	✓
	OPD (Agarwal et al., 2024)	✓	✗	✗	✗
	ReflectRL (Bi et al., 2026)	✓	✗	✓	✓
	Ours (VERA LLM)	✓	✗	✓	✓
Both	COS-PLAY (Wu et al., 2026b)	✓	✓	✗	✗
	Ours (VERA)	✓	✓	✓	✓

pass:

$$r_k = h_k \frac{\sum_j w_{k,j} z_{k,j}}{\sum_j w_{k,j}}, \qquad R_{\text{e2e}} = g\,[\lambda p + (1-\lambda)c], \tag{1}$$

where p is a process score, $c \in \{0,1\}$ a verified completion signal, $g \in \{0,1\}$ a trajectory-level hard gate, and λ the process weight. CoWork sets p to the mean of the five stage rewards, with λ=0.2 and g=1; MedResearch sets p to a judged process score, with λ=0.5 and g=0 upon verified reward hacking or a hard-gate failure (Appendix B). In words: a stage earns the weighted fraction of rubric items it passes, and nothing if a mandatory check fails; the end-to-end reward gives credit for partial progress but stays responsible for verified completion. Before a reward enters training, the *Reward Verifier* checks the rollout evidence behind it. Training focuses on the target stage from the LLM Report; neighboring stages and complete tasks stay in the mix, so stage-level gains must survive the full workflow. A candidate checkpoint that fails acceptance falls back to the previously accepted checkpoint. Unlike harness edits, model updates need not yield an immediate improvement. A candidate checkpoint is accepted if its development-benchmark score decreases by no more than 20% relative to the previously accepted checkpoint.

Update the harness, freeze the model. The Harness Report turns failed rubric items into candidate skill edits: add a missing skill, merge overlapping ones, or remove a harmful one (e.g., adding `early_stop` or merging `stage_tracker` and `plan_tracker` in Figure 1). The same LLM performs attribution and writes the proposed skill edits. Harness updates are restricted to skills; tools and execution rules remain fixed. The *Harness Verifier* tests each edit with paired runs on the development set: the same frozen model on the same tasks, with and without the edit, so only the edit can cause the difference. An edit is accepted only if it improves its target stage, causes no end-to-end regression, and passes safety checks; otherwise the current harness is kept. At most three edits are accepted per round. In addition, an edit must pass self-tests and improve the Overall development score by at least 5 % points. How often a skill is used is not evidence that it helps; only the paired comparison counts. Across rounds, the two axes feed each other: accepted skills change the next round's training distribution, and the improved model exposes the next harness weakness.

2.2 AUTO-RUBRIC

Rubrics A coding agent reads the task statement, the tool documentation, the execution logs, and the workspace, then writes a rubric: a list of binary checks, each scored in $\{0,1\}$, over action arguments, execution effects, artifacts, and output quality. Every check must be verifiable from observable evidence, so a rubric is at once the completion condition of a task and the evidence required to prove it.

Sandboxes As shown in Figure 3, we construct CoWork sandboxes from Terminal-Bench (Merrill et al., 2026), SWE-Bench (Jimenez et al., 2024), MLE-Bench (Chan et al., 2024), and medical

Table 2: **VERA achieves the best Overall and Pass@1 scores across both domains.** All methods use Qwen3.5-9B, with separate CoWork and MedResearch agents. O/A/T denote Overall/Agentic/Task, with Overall averaging Agentic and Task. SWE-Bench Verified (Jimenez et al., 2024) and MedXpertQA Text (Zuo et al., 2025) report Pass@1. Bold and underlined values indicate the best and second-best scores per column, respectively. Configuration details appear in Appendix C.5.

Method	AutoCoWorkBench			SWE-Bench Verified	AutoMedBench			MedXpertQA Text
	O	A	T	P@1	O	A	T	P@1
Base LLM + benchmark default harness								
Qwen3.5-9B	4.2	0.0	8.3	41.8	19.9	11.3	28.4	35.4
Qwen3.5-9B + Prompt	12.0	8.0	16.0	44.0	22.1	35.3	8.9	40.8
Harness Evolving								
ADAS (Hu et al., 2025)	<u>20.7</u>	22.3	19.1	52.4	55.5	61.6	<u>49.5</u>	42.0
Ours (VERA Harness)	18.2	21.3	15.1	<u>53.6</u>	<u>56.1</u>	<u>63.4</u>	48.8	42.7
LLM Evolving								
SFT (Ouyang et al., 2022)	6.7	1.7	11.7	42.0	12.9	21.6	4.1	32.1
GRPO (Shao et al., 2024)	5.8	5.0	6.7	14.8	36.3	47.0	25.7	22.3
RaR (Gunjal et al., 2025)	18.1	19.4	16.7	50.2	31.9	39.2	24.5	42.8
OPD (Agarwal et al., 2024)	19.8	15.6	**24.0**	49.2	41.2	51.8	30.6	44.5
ReflectRL (Bi et al., 2026)	17.3	20.1	14.5	48.4	28.6	20.7	36.5	40.9
Ours (VERA LLM)	18.9	<u>23.6</u>	14.2	51.2	43.3	50.6	36.1	<u>45.3</u>
Co-Evolving								
COS-PLAY (Wu et al., 2026b)	19.9	18.2	<u>21.6</u>	51.8	26.4	33.8	18.9	30.4
Ours (VERA Agent)	**31.0**	**45.3**	16.7	**54.2**	**69.1**	**76.8**	**61.3**	**50.8**

sandboxes from AutoMedBench (Liu et al., 2026b) and MedXpertQA (Zuo et al., 2025) workflows. Inspired by BaT's staged medical training (Liu et al., 2026a), we adopt five stages—*plan*, *setup*, *verify*, *execute*, and *submit*—to explore transfer across domains. A coding agent restores each stage's starting workspace, retains its query and interaction history, and attaches domain- and stage-specific rubrics grounded in task requirements and artifacts (Appendix A). Each stage can then be practiced and scored independently. End-to-end sandboxes retain the full workflow while an LLM surrogate simulates time-consuming tool calls (Ren et al., 2025). Both types undergo executable checks and Agent Judge verification before entering the training bank.

Verification Every candidate sandbox passes two kinds of checks. Executable code checks cover tool and skill schemas, restored states, and deterministic outputs. The Agent Judge, going beyond LLM-as-a-judge (Zheng et al., 2023), interacts with the workspace and checks arguments, execution effects, artifacts, and output context. A failed executable check or official verifier overrides any judge score. Sandboxes that fail are revised or rejected; only environments that run and score from evidence enter the training bank. This admission loop is what makes the corpus, and later the rewards, resistant to reward hacking.

3 Experimental Setup

We evaluate whether verified model–harness co-evolution improves long-horizon agents in CoWork and MedResearch. Our experiments compare learning methods, isolate each update axis, and assess scaling, transfer, and general-capability retention. **Domains and benchmarks.** Each domain uses separate agents and training sandboxes, with evaluation instances excluded from training. CoWork evaluation uses AutoCoWorkBench and SWE-Bench Verified; MedResearch uses AutoMedBench and MedXpertQA Text. Automation-Bench and AgentClinic assess transfer. Metrics and evaluation protocols appear in Appendix D. **Comparisons and scaling.** At 9B, we compare prompting, harness optimization, model training, and joint evolution under shared training data and matched fixed-

Table 3: **VERA-27B comes within 1.2 Overall points of the best frontier baseline in each domain.** O/A/T denote Overall/Agentic/Task. Bold and underlined values indicate the best and second-best scores per column, respectively. Agent and harness configurations are detailed in Appendix D.

AutoCoWorkBench				AutoMedBench			
Agent	**O**	**A**	**T**	**Agent**	**O**	**A**	**T**
Base LLM + benchmark default harness							
Qwen3.8-27B	63.8	60.8	66.7	Qwen3.8-27B	65.1	81.6	48.6
GPT-5.6-Sol	**72.6**	**76.8**	<u>68.3</u>	GPT-5.6-Sol	75.1	80.1	<u>70.0</u>
Claude-Opus-4.8	66.6	70.7	62.5	Claude-Opus-4.8	**81.9**	88.1	**75.8**
Base LLM + VERA Harness							
Qwen3.8-27B + VERA-Code	66.4	70.1	62.7	Qwen3.8-27B + VERA-Med	54.2	70.1	38.3
VERA LLM + benchmark default harness							
VERA-CoWork-27B	69.6	73.2	65.9	VERA-Med-27B	75.4	<u>91.7</u>	59.2
VERA Agent							
VERA-CoWork-27B Agent	<u>71.6</u>	<u>73.7</u>	**69.4**	**VERA-Med-27B Agent**	<u>80.7</u>	**94.1**	67.4

harness RL budgets (Appendix C). Harness-only and LLM-only variants at 9B and 27B isolate each component. We evaluate full co-evolution at 4B, 9B, and 27B, compare 27B agents with frontier models, and measure capability retention on AIME 2026, ALFWorld, GPQA-Diamond, and IF-Bench. **Implementation details.** We use Qwen3.5 at 4B and 9B and Qwen3.8 at 27B. Each run starts with SFT, followed by alternating model and harness updates. Training settings and execution configurations appear in Appendices B and D.

4 Results

4.1 VERA Agentic Co-Evolution Outperforms Single-Axis Optimization

VERA improves agent performance across both domains by jointly evolving the model and its harness (Table 2). It outperforms harness evolving with ADAS (Hu et al., 2025), LLM evolving training including GRPO (Shao et al., 2024) and RaR (Gunjal et al., 2025), and harness and llm joint evolution with COS-PLAY (Wu et al., 2026b). On AutoCoWorkBench, VERA-Code Agent reaches 31.0 Overall, exceeding the strongest single-axis baseline, ADAS (20.7), by 10.3 points. On AutoMedBench (Liu et al., 2026b), VERA-Med Agent reaches 69.1, exceeding its harness-only variant (56.1) by 13.0 points. The full method also leads on SWE-Bench Verified (Jimenez et al., 2024) and MedXpertQA Text (Zuo et al., 2025), with 54.2 and 50.8 Pass@1, respectively. These suites use official scoring protocols that differ from the training rubrics. Together, these results support the effectiveness of verified co-evolution across both stage-scored workflows and final-task evaluations.

Robustness The same loop drives both lines and improves both: one method, two tool ecosystems, no cross-domain interference, see Table 2. The robustness comes from the environments themselves, the same verifiable, restartable sandboxes feed the loop in both domains. The lines improve along different axes, harness and model contribute roughly equally on CoWork, while the harness arm carries more of the Med gain, so a full assessment needs the axis-level ablation and the scale analysis below.

Ablation Study. Freezing one axis costs about half the gain, see Table 2. On AutoMedBench, LLM-only reaches 43.3 and harness-only 56.1, against 69.1 for the full loop; on AutoCoWorkBench, 18.9 and 18.2 against 31.0. Every configuration shares the same sandboxes, budgets, and scorers, so each gap measures exactly the missing axis.

Table 4: **General-capability retention.** Parentheses show score changes from the corresponding backbone (green: gains; red: drops).

Model	AIME 2026	ALFWorld	GPQA-Diamond	IF-Bench
Qwen3.5-4B	89.7	32.1	76.2	59.2
VERA-CoWork-4B	33.3 (−56.4)	38.1 (+6.0)	64.3 (−11.9)	65.5 (+6.3)
VERA-Med-4B	30.0 (−59.7)	18.2 (−13.9)	61.1 (−15.1)	34.5 (−24.7)
Qwen3.5-9B	92.5	74.6	81.7	66.7
VERA-CoWork-9B	36.7 (−55.8)	53.2 (−21.4)	57.1 (−24.6)	59.8 (−6.9)
VERA-Med-9B	76.7 (−15.8)	68.4 (−6.2)	81.2 (−0.5)	52.0 (−14.7)
Qwen3.8-27B	86.7	94.0	82.2	79.5
VERA-CoWork-27B	96.7 (+10.0)	97.8 (+3.8)	89.9 (+7.7)	81.3 (+1.8)
VERA-Med-27B	90.0 (+3.3)	90.0 (−4.0)	87.8 (+5.6)	78.8 (−0.7)

4.2 VERA Scales to 27B and Reaches Frontier-Level Performance

At 27B, VERA approaches frontier performance (Table 3): 71.6 Overall on AutoCoWorkBench, above Claude-Opus-4.8 (66.6) and 1.0 point below GPT-5.6-Sol; 80.7 on AutoMedBench (Liu et al., 2026b), 1.2 points below Claude-Opus-4.8, with the highest Agentic score (94.1). Full co-evolution outperforms both single-axis variants in Overall score across both domains.

Figure 2 compares complete agent systems, including their harnesses, on additional workflows. VERA achieves 43.8% pass rate on Automation-Bench (Zapier, 2026), second among the compared agents, and 76.4% accuracy on AgentClinic (Schmidgall et al., 2024). Neither benchmark is used in sandbox construction, allowing us to assess transfer to new automation and clinical workflows.

4.3 Stronger Backbones Improve Agent Performance

Across AutoCoWorkBench and AutoMedBench (Liu et al., 2026b), VERA improves over its backbone at every tested scale, with Overall scores increasing from 4B to 9B to 27B (Figure 2). This comparison reflects both size and generation: Qwen3.5 supplies the 4B and 9B backbones, and Qwen3.8 supplies 27B.

General-capability retention varies by scale (Table 4). The 4B and 9B agents decline on several benchmarks, with AIME losses of 56.4–59.7 and 15.8–55.8 points, respectively. At 27B, the CoWork agent improves on all four benchmarks: AIME 2026, ALFWorld (Shridhar et al., 2021), GPQA-Diamond (Rein et al., 2023), and IF-Bench (Pyatkin et al., 2025). The Med agent improves on AIME and GPQA-Diamond, losing 4.0 points on ALFWorld and 0.7 on IF-Bench. Thus, 27B provides the strongest domain performance and capability retention among the tested backbones.

5 Related Work

5.1 Recursive Self-Improvement: Weights, Harness, or Both

Self-improvement requires reliable feedback. On the *weights* side, GRPO uses group-relative rewards (Shao et al., 2024), on-policy distillation learns from teacher feedback on student rollouts (Agarwal et al., 2024), and ReflectRL learns from negative trajectories (Bi et al., 2026). Agent Lightning separates execution from RL training (Luo et al., 2025), while Benchmark-as-Teacher (Liu et al., 2026a) uses medical benchmark rubrics (Liu et al., 2026b) for stage-wise GRPO training. On the *harness* side, methods improve reflection (Shinn et al., 2023), skill libraries (Wang et al., 2023), prompts and pipelines (Khattab et al., 2024; Agrawal et al., 2025), or agent workflows (Zhang et al., 2025b; Hu et al., 2025). Meta-Harness searches harness code using prior candidates and traces (Lee et al., 2026); the Darwin Gödel Machine evolves self-modifying agents (Zhang et al., 2025a); concurrent work studies harness-level RSI (Wu et al., 2026a). COS-PLAY jointly evolves a decision agent and skill bank (Wu et al., 2026b). Our focus is evidence-guided coordination of these two axes: VERA uses stage-wise rubric evidence to attribute failures, alternates model and harness updates,

and checks acceptance on a development set. We release 9,000+ training sandboxes to support further RSI research.

5.2 Verifiable Rewards and Judges

Reinforcement learning with verifiable rewards replaces a learned reward model with a programmatic checker, and has driven recent gains in reasoning (Lambert et al., 2024; DeepSeek-AI, 2025); process supervision adds verifiable intermediate signals along the solution path (Lightman et al., 2023; Wang et al., 2024). Outside domains with exact answers, rubrics and checklists break open-ended quality into itemized criteria that can serve directly as rewards (Gunjal et al., 2025; Viswanathan et al., 2025). Judging has evolved along a similar path, from LLM-as-a-judge (Zheng et al., 2023) to agent-as-a-judge, where an agent inspects the workspace and intermediate steps of another agent instead of reading only its final answer (Zhuge et al., 2024). Any reward produced by a model, however, can be over-optimized by the policy it trains (Gao et al., 2023). Existing rubric rewards score single responses, and agent judges have so far been used for evaluation rather than for training. VERA writes rubrics per stage of a long-horizon trajectory, grounds each check in an agent judge that inspects the workspace, and lets a failed executable check override any model judgment, so the same rubrics can serve as both an admission gate and a training reward.

5.3 Environments for Training Agents

Agent benchmarks now ship realistic task suites with official checkers: SWE-Bench Verified (Jimenez et al., 2024), Terminal-Bench (Merrill et al., 2026), MLE-Bench (Chan et al., 2024), τ-Bench (Yao et al., 2025), and domain suites for web (Zhou et al., 2024) and medical (Schmidgall et al., 2024; Zuo et al., 2025) agents. These are static evaluation sets: they score the final outcome of a full episode and cannot be restarted at an intermediate stage. A parallel line scales executable environments for training coding agents, collecting or synthesizing repository tasks at scale (Pan et al., 2024; Jain et al., 2025; Yang et al., 2025); success there is defined by the repository's test suite, which ties the approach to domains with tests and rewards only the end of an episode. VERA instead converts benchmark trajectories from both coding and medical research into sandboxes that restart at any of five stages, attaches a verified rubric to each stage, and admits a sandbox only if it runs and scores from observable evidence.

6 Limitations

Our study leaves five aspects for further investigation. **(1) Environment scaling:** We do not vary environment counts or compare construction methods. **(2) Harness ablations:** Runtime settings, including reasoning effort, are fixed and not ablated. **(3) Judge dependence:** Judging, verification, and attribution rely on GPT-5.6-Sol; executable checks constrain its judgments, but sensitivity to judge choice remains untested. **(4) Baseline scale:** Full baseline comparisons are limited to 9B; a 27B co-evolution run costs approximately 5,000–8,000 GPU-hours. **(5) Comparison scope:** We compare learning algorithms rather than alternative operators within VERA. Rubrics-as-Rewards remains fixed in our loop; we do not separately ablate alternative reward formulations or update operators.

7 Conclusion

We introduced VERA, a framework for recursive self-improvement through verifiable environments and model–harness co-evolution. Auto-Rubric converts task trajectories into restartable sandboxes with evidence-grounded feedback. This feedback identifies failures, guides alternating model and skill updates, and supports verification of proposed changes.

Across CoWork and MedResearch, co-evolution outperforms updating either component alone, with further gains at larger model scales. By releasing 9,000+ verifiable environments, we provide a foundation for studying how agents can learn from their own attempts. Future work can extend this approach to broader workflows and examine how environment scale and verification quality affect sustained self-improvement.

AI USE STATEMENT

We used generative AI tools to clean and reformat writings, create and edit software code, create and modify scientific figures and images, and improve the readability of the manuscript. The authors take responsibility for the final content and all AI-assisted contributions.

ETHICS STATEMENT

Our evaluation uses existing benchmarks and reference materials; it does not establish clinical safety or effectiveness. The system may reproduce errors and biases in models or reference data. Any clinical use would require independent validation, privacy safeguards, and qualified human oversight.

REPRODUCIBILITY STATEMENT

The appendix provides the implementation details, including data splits, reference sources, input limits, generation settings, and the safeguards against answer leakage and circular validation.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes, 2024. URL https://arxiv.org/abs/2306.13649.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Jinhe Bi, Chennan Zhou, Zengjie Jin, et al. ReflectRL: Learning from golden negative trajectories via reflective-to-direct reasoning. *arXiv preprint arXiv:2608.03972*, 2026.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024. URL https://arxiv.org/abs/2410.07095.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, 2023.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *arXiv preprint arXiv:2504.07164*, 2025. URL https://arxiv.org/abs/2504.07164.

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 *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, et al. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, et al. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. URL https://arxiv.org/abs/2305.20050.

Junqi Liu, Yufan He, Yexiao He, Pengfei Guo, Dong Yang, Andriy Myronenko, Can Zhao, Hanrong Ye, Tianhao Qi, Yuyin Zhou, Daguang Xu, and Yucheng Tang. BaT: Towards self-evolving medical research agent with stage rubrics. *arXiv preprint arXiv:2608.16211*, 2026a.

Junqi Liu, Selena Song, Yuhan Wang, Jiawei Mao, Hardy Chen, Xiaoke Huang, Tianhao Qi, Pengfei Guo, Yucheng Tang, Yufan He, Can Zhao, Andriy Myronenko, Dong Yang, Daguang Xu, and Yuyin Zhou. AutoMedBench: Towards medical autoresearch with agentic AI models. *arXiv preprint arXiv:2606.01961*, 2026b.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train ANY AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025. URL https://arxiv.org/abs/2508.03680.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Brovych Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations (ICLR)*, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, et al. Training software engineering agents and verifiers with SWE-Gym. *arXiv preprint arXiv:2412.21139*, 2024.

Valentina Pyatkin et al. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2025.

David Rein et al. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Zhenzhen Ren, Xinpeng Zhang, Zhenxing Qian, Yan Gao, Yu Shi, Shu-Xin Zheng, and Jiyan He. Gtm: Simulating the world of tools for ai agents. *ArXiv*, abs/2512.04535, 2025. URL https://api.semanticscholar.org/CorpusID:283556867.

Samuel Schmidgall et al. AgentClinic: A multimodal agent benchmark to evaluate AI in simulated clinical environments. *arXiv preprint arXiv:2405.07960*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Nazari Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, et al. Checklists are better than reward models for aligning language models. *arXiv preprint arXiv:2507.18624*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9053–9070, 2024.

Siwei Wu, Jincheng Ren, Yizhi Li, Haau-Sing Li, Chengran Yang, et al. ModularRSI: Modular and generalizable recursive harness self-improvement. *arXiv preprint arXiv:2609.14857*, 2026a.

Xiyang Wu, Zongxia Li, Guangyao Shi, Alexander Duffy, Tyler Marques, Matthew Lyle Olson, Tianyi Zhou, and Dinesh Manocha. Cos-play: Co-evolving llm decision and skill bank agents for long-horizon tasks. *arXiv preprint arXiv:2604.20987*, 2026b.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, et al. SWE-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Zapier. AutomationBench: Realistic business workflow automation via REST APIs. https://github.com/zapier/AutomationBench, 2026.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025a.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, et al. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025b.

Zili Zhang, Yinmin Zhong, Chengxu Yang, Chao Jin, Bingyang Wu, Xinming Wei, Yuliang Liu, and Xin Jin. Heddle: A distributed orchestration system for agentic RL rollout. *arXiv preprint arXiv:2603.28101*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, et al. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. MedXpertQA: Benchmarking expert-level medical reasoning and understanding. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.