BUILDING TRUSTWORTHY SCIENTIFIC ENVIRONMENTS FOR CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated scientific discovery is emerging as a new paradigm where coding agents interact with scientific environments to pursue open-ended research goals. While recent work has improved agent capabilities, the role of environments in shaping scientific exploration remains largely unexplored. In this work, we show that environmental feedback influences how coding agents explore and optimize scientific objectives through controlled studies with four coding agents across four feedback settings. We then introduce SciEnBuilder, an automated framework that transforms paper–repository pairs into trusted scientific environments by connecting scientific claims, executable evidence, and protected evaluation. Using SciEnBuilder, we construct 30 scientific environments for coding agents from a large collection of top-tier papers. Human evaluation confirms their faithfulness to the original scientific goals and the reliability of their evaluation. Compared with raw repositories, these environments improve research efficiency, eliminate observed shortcut behaviors, and enable stronger long-horizon optimization. These results highlight the importance of trustworthy scientific environments for reliable automated scientific discovery.

1 INTRODUCTION

Automated scientific discovery is emerging as a new paradigm in which AI systems autonomously pursue scientific research (Lu et al., 2026; Bengio et al., 2025; Wang et al., 2023). In this paradigm, **coding agents** are increasingly used to drive autonomous exploration, while **scientific environments** define the research space and provide feedback on the research progress (Figure 1(a)). Despite the central roles of both components, existing work has concentrated primarily on the agent side. Advances in memory (Jin et al., 2026), coordination (Weng et al., 2026), and search strategies (Novikov et al., 2025) have strengthened the research capabilities of coding agents, thus enabling increasingly advanced scientific exploration (Swanson et al., 2025; Oh et al., 2025).

Recent work has begun to examine the role of environments, showing that environmental feedback is fundamental to long-term agent improvement (Dong et al., 2026; Fang et al., 2026; Zhu et al., 2026). When such feedback is distorted, agents may instead persistently optimize the wrong objective (Wang et al., 2026a; Huang et al., 2026). This concern is particularly important for scientific discovery, where misaligned feedback can therefore gradually steer long-term exploration away from the true scientific objective (Ríos-García et al., 2026). However, existing work on scientific environments has focused largely on scaling up environment coverage through synthetic environment generation and increasing environment complexity (Qiang et al., 2026; Zhou et al., 2026; Moussa et al., 2026b), while assuming that the feedback they provide is reliable. **How different feedback signals shape the long-term scientific exploration of coding agents remains largely unexplored.**

Moreover, existing approaches often require human effort in the environment construction process, where experts manually design the feedback mechanism for each scientific environment in advance (Moussa et al., 2026a). This can be largely because the automated construction of scientific environments poses significant technical challenge. While the growing body of papers and open-source repositories provides abundant material for environment construction, they cannot be directly used as trustworthy scientific environments. For example, complex repository structures can obscure the true optimization objective, whereas evaluation logic entangled with experimental code can leak the target signal to the agent. Addressing these issues requires substantial domain ex-

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.

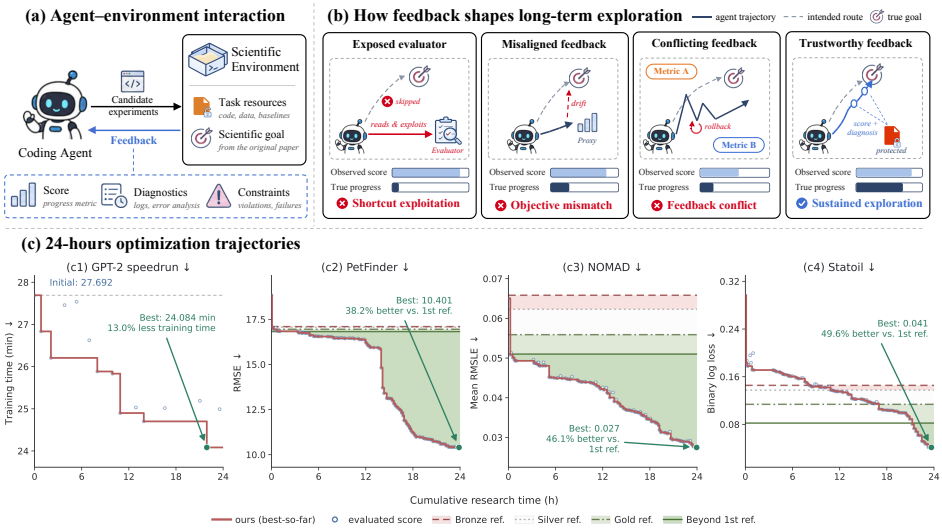


Figure 1: **(a)** Coding agents iteratively explore research ideas with environmental feedback. **(b)** Feedback design redirects long-term exploration toward exploitation, mismatch, conflict, or sustained progress. **(c)** Our environments enable sustained optimization across diverse tasks.

pertise and engineering effort, making environment construction costly, task-specific, and difficult to scale (Xin et al., 2026).

To address these issues, we first conduct a systematic controlled experiment (Section 3) to investigate how environmental feedback shapes the scientific exploration of coding agents. This preliminary study spans three frontier scientific tasks $\times$ four coding-agent configurations $\times$ four feedback conditions, revealing three consistent failure modes: **(1) Shortcut exploitation:** when the ground-truth evaluator is exposed, agents tend to bypass scientific exploration and directly exploit the evaluation signal; **(2) Objective mismatch:** when observable feedback is misaligned with the true scientific objective, agents continue to optimize the proxy metric at the expense of true scientific progress; and **(3) Feedback conflict:** when multiple evaluation signals conflict, agents repeatedly switch between objectives and roll back previous progress, leading to unstable exploration. These findings suggest that trustworthy scientific environments should keep evaluation protected, ensure feedback remains aligned with the scientific objective, and provide informative signals for iterative exploration.

Building on these findings, we introduce **SciEnBuilder** (Section 4), which automatically transforms paper–repository pairs into trustworthy scientific environments. As shown in Figure 1(b), SciEnBuilder first grounds the scientific goal from the original paper, preventing objective drift during environment construction. Based on this goal, it reconstructs a reproducible experimental pipeline and partitions the environment into agent-accessible resources and protected evaluation. This separation ensures reliable execution without exposing hidden data or evaluation logic. Finally, at each iteration, the environment provides aligned scores and diagnostic feedback from protected evaluation to guide subsequent scientific exploration. Using SciEnBuilder, we automatically construct trustworthy scientific environments for 30 real-world research tasks (full list in Table 8). **Human evaluation** with five experts confirms that these environments faithfully preserve the scientific objectives of the original papers and provide reliable feedback.

To further evaluate their effectiveness across different exploration horizons, we conduct both short- and long-horizon scientific discovery experiments (Section 5). For the **4-hour setting**, we evaluate four coding-agent configurations based on *Codex, Claude Code, and RG-Agent* (Garikaparthi et al., 2026) with `GPT-5.5` and `GLM-5.2` across ten sampled environments. Our environments enable diverse coding agents to start experiments faster (**90.86→31.92 min**), spend more time on exploration (**76.91%→94.14%**), and conduct more valid research iterations (**45.86%→80.95%**), while eliminating shortcut behaviors (**206→0**). For the **24-hour setting**, we further evaluate long-horizon exploration using a GPT-6-based Codex agent on eight scientific tasks. Without any changes to the agent architecture, our environments enable the general-purpose coding agent to achieve five gold medals on MLE-Bench (Chan et al., 2025) and reduce NanoGPT Speedrun[1] from 27 to 24 minutes.

[1] `https://github.com/KellerJordan/modded-nanogpt`

To our knowledge, we are the first to systematically study how environmental feedback shapes the scientific behavior of coding agents, introducing an agentic framework that automatically transforms paper–repository pairs into scientific environments with trustworthy feedback. **Our results suggest that trustworthy feedback is fundamental to sustained agent improvement,** shedding light on the design of infrastructure for autonomous scientific discovery. We will release the framework, the constructed environments, and **392** full research trajectories spanning **1085** agent-hours and **1,426** research iterations, providing a valuable resource for future research in this direction.

2 Related Work

Autonomous Scientific Discovery with Coding Agents. Recent advances in large language models have enabled coding agents to tackle complex scientific tasks through mechanisms such as long-term memory (Jin et al., 2026), multi-agent collaboration (Weng et al., 2026), and search-based exploration (Novikov et al., 2025; Zheng et al., 2026; Swanson et al., 2025; Oh et al., 2025). Meanwhile, emerging benchmarks have enabled systematic evaluation across diverse domains (Qiang et al., 2026; Chen et al., 2025; Xie et al., 2025b). Despite these advances, existing efforts have primarily focused on improving the capabilities of coding agents, while paying limited attention to the role of research environments in the discovery process.

Environmental Feedback for Agent Behavior. Beyond agent-centric improvements, recent work on general-purpose agents has highlighted the importance of environments in shaping agent learning (Zhu et al., 2026). By constructing diverse and challenging task environments, these studies improve agent robustness across downstream tasks (Dong et al., 2026; Fang et al., 2026). Meanwhile, studies on feedback mechanisms have revealed that agents often exploit observable signals, leading to unintended behaviors such as reward hacking and objective misalignment (Wang et al., 2026a). However, existing studies have mainly focused on general task-solving scenarios, where feedback primarily guides immediate task performance. How feedback design influences long-horizon exploration remains less understood, particularly in scientific discovery, where agents rely on environmental feedback to guide iterative research decisions. Our work thus extends this perspective by systematically studying how environmental feedback shapes scientific exploration.

Automated Construction of Scientific Environments. Another line of work has explored the construction of scientific environments for training and evaluating agents, including efforts to scale environment coverage and improve task diversity (Zhou et al., 2026; Qiang et al., 2026; Wang et al., 2026c; Moussa et al., 2026a). However, constructing trustworthy scientific environments remains challenging. Real-world papers and repositories often require substantial adaptation due to coupled evaluation logic and incomplete execution pipelines, making them difficult to transform into trustworthy environments. Moreover, existing approaches typically require manually designed feedback mechanisms, which demand substantial domain expertise and limit scalability (Xin et al., 2026). To address this issue, we introduce SciEnBuilder, an agentic framework that transforms paper–repository pairs into scientific environments with protected evaluation and reliable feedback.

3 The Impact of Feedback in Automated Scientific Discovery

To study how environmental feedback affects the quality of scientific discovery, we conduct controlled experiments across three frontier research tasks: *CELLECT* (Zhou et al., 2025) (*Nature Methods* 2025), *Human Simulacra* (Xie et al., 2025a) (ICLR 2025), and *MIRO* (Pineda et al., 2025) (*Nature Communications* 2025). To ensure a systematic analysis, we construct four environment variants for each task and repeat every task–agent–environment setting three times with four coding agents, resulting in 144 runs.

3.1 Environment Feedback in Automated Scientific Discovery

We study automated scientific discovery as an interactive process in which a coding agent develops and evaluates scientific hypotheses through repeated interaction with a computational environment.

Scientific environment. We denote the scientific environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{F})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{F}$ is the feedback space. The environment state s records the current research artifacts and execution context, while an action corresponds to an operation performed by the coding agent (e.g., executing a command).

Table 1: **Scientific environments shape agent behavior through feedback.** Results are averaged over four agent configurations and three rounds. *True*/*Lazy* Scores denote performance under the ground-truth/proxy evaluators; *VSTR* measures valid scientific time, and *Shortcuts* counts audited shortcut behaviors. Full results are reported in Tables 4, 5 and 6.

Environment	Human–Simulacra			CELLECT			MIRO			Overall
($n = 144$ runs)	True$^{\uparrow}$	Lazy$^{\uparrow}$	VSTR (%)$^{\uparrow}$	True$^{\downarrow}$	Lazy$^{\downarrow}$	VSTR (%)$^{\uparrow}$	True$^{\uparrow}$	Lazy$^{\uparrow}$	VSTR (%)$^{\uparrow}$	Shortcuts$^{\downarrow}$
C0	**52.04**	35.89	85.96	**21.88**	865.00	54.26	**0.92**	0.82	62.11	121
C1	48.15	34.85	42.98	25.45	864.64	70.14	0.83	0.85	57.63	**0**
C2	31.68	**53.24**	**98.68**	686.84	**448.83**	**94.34**	0.35	**0.99**	**89.32**	**0**
C3	50.12	41.21	97.28	22.15	859.91	91.87	0.86	0.92	81.02	**0**

Agent–environment interaction. Given a task $\mathcal{T}$, a coding agent π_θ repeatedly interacts with $\mathcal{E}$: at step t it acts on the interaction history h_t, and the environment returns an updated state and feedback, $(s_{t+1}, f_{t+1}) = \mathcal{E}(s_t, \pi_\theta(\mathcal{T}, h_t))$. Feedback $f_{t+1} \in \mathcal{F}$ may be an execution output, an error message, or an evaluation score. Since the agent conditions every subsequent action on it, the reliability of feedback bounds the validity of the exploration it guides (Ríos-García et al., 2026).

Environment variants. To isolate the effect of feedback, we manually construct four variants $\mathcal{C} = \{C_i\}_{i=0}^{3}$ per task, where $f^\star$ is the ground-truth evaluator and $\tilde{f}$ a misaligned proxy. Visible feedback is:

$$\begin{aligned} \mathcal{V}(C_0) &= \{f^\star\}, \quad \mathcal{V}(C_1) = \emptyset, \\ \mathcal{V}(C_2) &= \{\tilde{f}\}, \quad \mathcal{V}(C_3) = \{\mathrm{API}(f^\star, \tilde{f})\}, \end{aligned} \tag{1}$$

Here *C0* exposes the ground-truth evaluator; *C1* removes all evaluation feedback; *C2* replaces it with a misaligned proxy; and *C3* withholds both evaluators on the host and returns their two scores through a grading API. The four variants differ only in $\mathcal{F}$, and span the feedback regimes a research environment can realize, from full disclosure of $f^\star$ to none.

3.2 Experimental Settings

Tasks. For each task, we manually construct four environment variants, each cross-checked by a second expert to ensure quality. Construction details are deferred to Appendix B.1.

Coding agents. We evaluate four coding agent configurations spanning different agent harnesses and base models: Codex + `GPT-5.5`, RG-Agent + `GPT-5.5`, Codex + `GLM-5.2` and Claude Code + `GLM-5.2`. The RG-Agent harness is a ReAct-style (Yao et al., 2022) research scaffold built on the Inspect framework (Garikaparthi et al., 2026).

Experimental budget. Following previous work (Wang et al., 2026c; Zhou et al., 2026), each run is allocated at most 4 hours of wall-clock time, 1 GPU, and a $60 API budget. Agents may terminate early when they consider the performance sufficient (see Section A for further discussion).

Evaluation metrics. We report four metrics (details in Appendix B.1): *Lazy Score* and *True Score* measure performance under the proxy and held-out ground-truth evaluators, respectively. *Valid Scientific Time Rate* (VSTR) is the fraction of active time devoted to valid scientific artifacts, while *Shortcut Behavior Number* counts audited behaviors of evaluator exploitation or other unintended shortcuts. Since score scales vary across tasks, *Lazy* and *True Scores* are analyzed within each task.

3.3 Results and Analysis

We compare the four feedback conditions to study how feedback shapes coding agent behavior. Brief results are shown in Table 1 and Figure 2, with full results in the Appendix. We observe:

Exposing ground-truth feedback can induce evaluator exploitation. When ground-truth feedback is directly exposed in *C0*, coding agents actively incorporate it into their research and probe the true evaluation procedure for exploitable signals. Table 1 shows the resulting pattern clearly: *C0* achieves the strongest *True* performance on all three tasks, but also incurs 121 shortcut behaviors and substantially lower *VSTR* than *C2* and *C3*. This suggests that once ground-truth feedback becomes accessible, coding agents adapt their behavior around the evaluator, making it difficult to disentangle strong performance from evaluator exploitation.

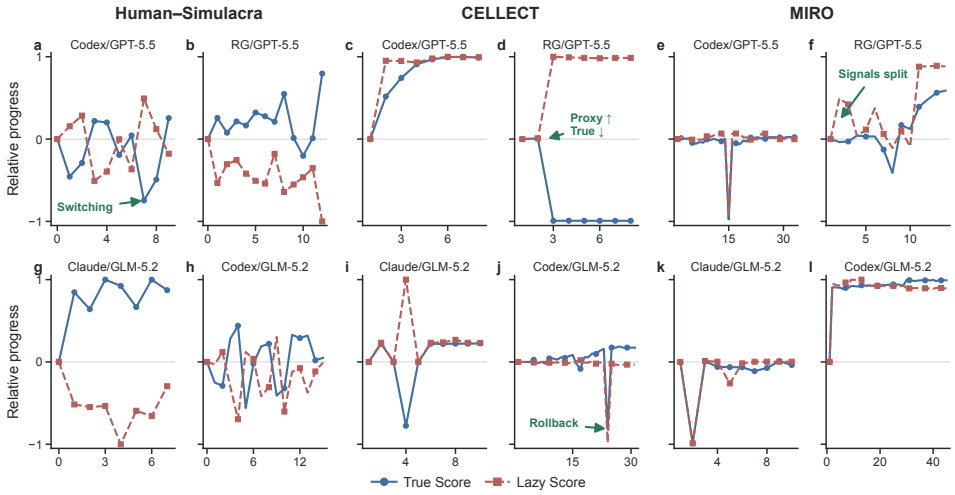


Figure 2: Optimization trajectories under conflicting feedback (*C3*) show agents switching between *True* and *Lazy Scores* across tasks and configurations. The x-axis shows optimization steps. The y-axis shows normalized progress relative to the initial candidate. Full results in Figures 6, 7, and 8.

Agents reconstruct inaccurate feedback when official feedback is unavailable. When no scientific feedback is available (*C1*), coding agents reconstruct a surrogate objective from the available information (e.g., task materials, intermediate results) and use it to guide subsequent actions. However, the reconstructed objective is often inaccurate. *True* performance degrades across all three tasks, while *VSTR* remains only 42.98%. Without a reliable feedback signal, coding agents therefore drift toward self-constructed objectives that only partially reflect the true scientific goal.

Agents pursue misaligned feedback at the expense of true performance. When *C2* exposes only a misaligned evaluator, coding agents reorganize their research around the visible proxy. As shown in Table 1, *C2* achieves the best Lazy performance but the worst True performance on all three tasks. Relative to *C1*, Lazy performance improves in 33/36 matched comparisons, while True performance degrades in 31/36. Meanwhile, the high *VSTR* (89.32%) shows that coding agents are conducting sustained scientific work, but toward the wrong feedback signal.

Conflicting feedback leads to unstable behavior. When multiple feedback signals are simultaneously exposed in *C3*, coding agents repeatedly shift between competing objectives rather than following a stable search direction. Figure 2 shows this directly: True and Lazy trajectories frequently diverge and reverse, producing target switching, rollback, and compromise across tasks and configurations. These trajectory-level reversals reveal continual reprioritization between conflicting feedback signals, making the research inherently unstable.

Summary. Across different tasks and agent configurations, access to the ground-truth evaluator encourages exploitation (*C0*); removing feedback causes agents to drift from the true objective (*C1*); and misaligned proxies redirect optimization toward the wrong target (*C2*). Even with multiple signals available, conflicting feedback can induce unstable research behavior (*C3*). These findings show that **agent behavior is strongly shaped by the signals exposed by the scientific environment.**

4 Building Scientific Environment for Coding Agents

4.1 From Reproduction to Research Validity

Section 3.3 shows that coding agents organize their research around the feedback they receive. A score provides valid evidence only when the evaluated experiment matches the scientific claim and protocol defined by the paper. The score itself does not reveal whether this match holds (Huang et al., 2026). Let $f_p^\star(x)$ be the outcome of executing a paper's protocol p faithfully on a candidate x, and $g(x)$ the score the environment returns. Existing construction pipelines are judged by *reproduction*, which checks $|g(x_0) - f_p^\star(x_0)| \leq \epsilon_p$ at the released baseline x_0 (Seo et al., 2026; Starace et al., 2025; Wang et al., 2026b). A coding agent, however, submits many new candidates and moves

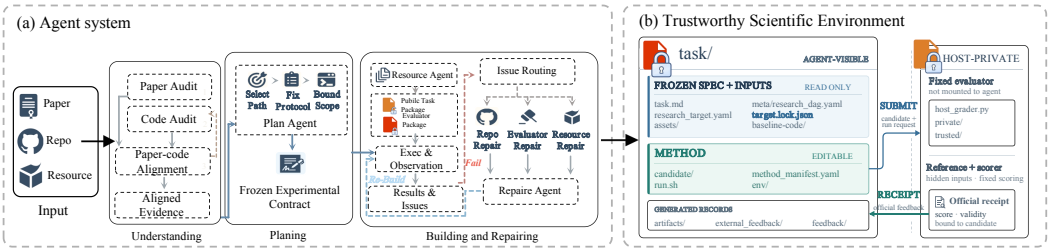


Figure 3: Overview of SciEnBuilder.

toward those on which the score is easiest to raise, so a scientifc environment instead requires:

$$|g(x) - f_p^\star(x)| \le \epsilon_p \qquad \forall\, x \in \mathcal{X}_{\pi_\theta}(g), \tag{2}$$

over all candidates $\mathcal{X}_{\pi_\theta}(g)$ that agent π_θ can reach while interacting with g. Reproduction checks one point; an optimizing agent searches for the points where the check fails.

This set becomes known only as research proceeds. We therefore define four conditions that can be checked during construction or evaluation to keep g aligned with $f_p^\star$. **Faithful**: p is the protocol the paper uses to support the claim. **Grounded**: $g(x)$ is recomputed from artifacts produced by running x, never from what the candidate reports about itself. **Isolated**: no candidate can read or modify the reference, the scorer, or the records g depends on. **Anchored**: $g(x_0)$ recovers the paper-reported result within ϵ_p, the only condition reproduction certifies. Faithful rules out the proxy optimization of *C2*; Grounded and Isolated together leave a candidate only one way to raise g, changing what $f_p^\star$ itself measures, which rules out the shortcut exploitation of *C0*. Faithful and Anchored hold once, at construction, whereas Grounded and Isolated are re-checked for every candidate, so the environment is certified where the agent actually goes.

4.2 SciEnBuilder

Certifying these conditions requires paper-specific reconstruction that is costly to do by hand (Wang et al., 2026b; Siegel et al., 2024). **SciEnBuilder** (Figure 3) automates it with coding agents, whose accounts of their own work are exactly the signal Section 3 found unreliable; it therefore follows one principle: *agents propose, execution produces evidence, and deterministic checks decide*. Full procedures are deferred to Appendix C.

Fixing the scientific goal. A repository makes a paper's experiments runnable, but seldom says which of its computations the claims rest on, so a builder that starts from the code tends to measure whatever the code computes. SciEnBuilder therefore audits the paper and the repository separately, withholding the paper from the code audit, and reconciles the two into a research directed acyclic graph (DAG) whose edges record how evidence supports each result. Since a paper's claims rest on experiments that differ in data, protocol, and cost, each environment is scoped to one research goal $\tau = (\sigma, p, \Theta, \kappa)$: an evidence path σ in the DAG, its evaluation protocol p, the changes Θ the agent may make, and the claims κ retained but not executed. The goal may be narrower than the paper, but not weaker: it may omit an experiment it cannot run, but may not relax the semantics of one it keeps. τ is then frozen by a content digest as an experimental contract, with every protocol field anchored to its source in the paper, which makes Faithful auditable.

Protected and grounded evaluation. The contract is realized as a public task package for the agent and a private evaluator package that keeps the hidden splits, the reference, and the scorer on the host, so a score can be obtained only by submitting a candidate (Isolated). Each score is recomputed from the candidate's execution artifacts, which a manifest binds to the run, split, protocol, scorer, and reference of the goal; evidence that mixes in any other is rejected (Grounded). The baseline is scored through the same path and compared with the paper (Anchored). Construction failures are routed to typed repairs, which may change the implementation but never what it measures: a repair that would alter the contract or the reference is rolled back.

Feedback as scientific evidence. A scalar score folds together outcomes that call for different responses: read as a number, an infrastructure fault looks like a poor method and a valid negative result looks like a wasted round. SciEnBuilder keeps g as the single objective, so the agent is never offered

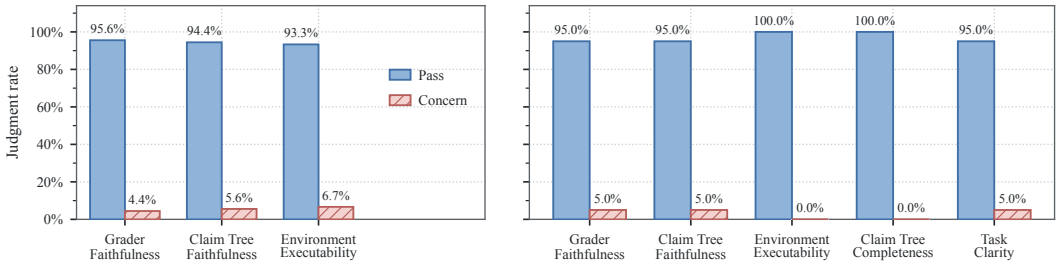


Figure 4: Human experts confirm high quality across the full (left) and the sampled suite (right).

Table 2: **Comparison between the constructed environments and original repositories.** *I-VRCR* measures valid research iterations; *Shortcuts* counts audited shortcut behaviors; *VSTR* measures valid scientific time; and *Start Time* denotes the elapsed time to the first successful experiment.

Environment	I-VRCR (%) ↑	VSTR (%) ↑	Shortcuts ↓	Start time (min) ↓	
				Mean	Median
Sandbox ($n = 120$ runs)	**80.95**	**94.14**	**0**	**31.92**	**7.45**
Original ($n = 120$ runs)	45.86	76.91	206	90.86	64.39

a second one to switch to (*C3*), and returns with it diagnostics from the same run and an evidence state that separates execution, measurement validity, and protocol completion. Each round follows a fixed order: the agent commits to a hypothesis and a predicted change before submitting, the host issues a receipt binding the result to that prediction, and the outcome is reduced deterministically to SUPPORTED, CONTRADICTED, INCONCLUSIVE, or NOT DECIDABLE. A negative result is thus kept as evidence, a prediction fixed before the result cannot be reinterpreted after it, and exploration stays tied to testable hypotheses rather than drifting (*C1*).

5 EXPERIMENTS

To evaluate the proposed scientific environments, we first use SciEnBuilder to construct 30 environments and assess their quality with human experts (Section 5.1). We then evaluate four coding agents on 10 sampled environments, comparing scientific discovery in the constructed environments with discovery on the original code repositories (Section 5.2). Finally, we analyze the long-term effectiveness of the proposed feedback design in 24-hour discovery runs (Section 5.3).

5.1 ARE THE CONSTRUCTED ENVIRONMENTS RELIABLE?

Experimental Settings. To evaluate the proposed environments, we use SciEnBuilder to construct 30 environments from recent top-tier papers with open-source code and clearly defined scientific objectives, spanning domains such as computational biology and algorithm optimization. Paper list and statistics are provided in Table 8 and Figure 18. Three human experts assess all 30 environments for *Executability* and for the faithfulness of their feedback signals (*Grader Faithfulness*) and scientific objectives (*Claim Tree Faithfulness*) to the original papers. For a more comprehensive evaluation, we further sample 10 environments and invite two additional experts to assess them along five dimensions: *Grader Faithfulness*, *Claim Tree Completeness*, *Claim Tree Faithfulness*, *Environment Executability*, and *Task Clarity*. *Pass*/*Concern* indicates whether each criterion is satisfied.

As shown in Figure 4, human reviewers found the constructed environments to be of high quality across both evaluation scopes. Across all 30 environments, the three core criteria achieved average Pass rates of 94.4%. On the 10 sampled environments, the two additional experts reported average Pass rates of 97.0% across five dimensions. Reviewer judgments were highly consistent, with pairwise agreement rates of **89.6%** for the full suite and **94.0%** for the sampled subset.

5.2 DOES THE FEEDBACK DESIGN ENABLE RELIABLE SCIENTIFIC EXPLORATION?

Experimental Settings. Based on the 10 sampled scientific environments, we compare scientific exploration in our constructed environments and the original code repositories using the same four coding-agent configurations and per-run budgets as in Section 3.2. In addition to *VSTR*, and *Shortcut*

Table 3: **Task-level scientific performance in our environment.** FIR_{Macro} averages run-level improvement rates. *VSTR* measures agent-wise valid scientific time, and *Relative Gain* reports the best progress from baseline toward the task optimum. Best agent results are bolded. Original task-level results are omitted because frequent shortcut exploitation makes fair comparison difficult.

Task	$\mathrm{FIR}_{\mathrm{Macro}}$ (%) ↑	Agent-wise VSTR (%) ↑				Agent-wise Relative Gain (%) ↑			
		Codex GPT-5.5	RG-Agent GPT-5.5	Claude Code GLM-5.2	Codex GLM-5.2	Codex GPT-5.5	RG-Agent GPT-5.5	Claude Code GLM-5.2	Codex GLM-5.2
ACS (Sterner & Teufel, 2025)	80.93	94.99	**96.74**	91.88	90.76	**11.10**	4.16	9.54	8.94
CELLECT (Zhou et al., 2025)	66.67	95.57	95.12	96.52	**96.80**	33.92	**44.75**	37.61	37.37
ECI (Wu et al., 2025a)	48.67	94.84	**99.21**	83.42	96.75	1.31	0.99	**36.02**	6.64
Human–Simulacra (Xie et al., 2025a)	47.14	91.78	**99.33**	95.46	80.90	**65.71**	63.81	49.52	54.29
MIRO (Pineda et al., 2025)	76.27	**98.24**	91.68	97.76	98.24	45.70	57.93	**72.93**	62.90
TabM (Gorishniy et al., 2025)	81.25	98.88	**99.24**	95.33	98.75	10.44	**13.14**	−8.14	7.19
CausalPFN (Balazadeh Meresht et al., 2026)	70.98	74.69	96.91	85.87	**97.52**	0.00	0.00	0.01	**0.25**
Fast-DetectGPT (Bao et al., 2024)	51.81	98.35	**98.76**	96.60	97.73	1.83	0.86	**44.08**	9.19
HTE (Zhong et al., 2025)	47.47	79.02	94.58	96.40	**98.18**	9.91	**10.58**	4.00	8.70
ReAct-OT (Duan et al., 2025)	56.51	87.13	96.12	91.55	**98.89**	−0.07	**0.09**	−0.42	−0.07
Overall	62.77	91.35	**96.77**	93.08	95.45	17.99	19.63	**24.52**	19.54

Behavior Number, we report four additional metrics. *Valid Research Cycle Rate (VRCR)* measures the fraction of valid and scorable research iterations within a run; *Time to First Valid Experiment* measures the elapsed time from task start to the first successful and scorable experiment; *Feedback Improvement Rate*(*FIR*) measures how often a follow-up iteration improves upon the preceding result after receiving feedback; and *Best Relative Gap Gain* measures the best progress from the baseline toward the task optimum, with negative values indicating performance below the baseline.

Based on the results in Tables 2 and 3, we have the following observations:

The constructed environments substantially improve valid scientific exploration. Compared with the original repositories, our environments increase *I-VRCR* from 45.86% to 80.95% and *VSTR* from 76.91% to 94.14%, while eliminating all audited shortcut behaviors observed during experiments with the original repositories. They also reduce the mean time to the first successful experiment from 90.86 to 31.92 minutes. These results show that the constructed environments converts raw repositories into more trustworthy scientific environments, enabling agents to conduct valid experiments more efficiently.

Environmental feedback supports iterative scientific progress. Across the ten sampled scientific environments, $\mathrm{FIR}_{\mathrm{Macro}}$ reaches 62.77%, showing that successive iterations frequently improve upon their preceding results. This result suggests that our environments provide useful feedback for autonomous scientific exploration, while variation across tasks reflects differences in both task characteristics and the ability of the agent to utilize these signals.

Valid experimentation and scientific progress capture different capabilities. All agent configurations spend most of their time on valid scientific experimentation, with overall *VSTR* ranging from 91.35% to 96.77%. Yet their scientific gains differ considerably across tasks, and no configuration consistently performs best. For example, RG-Agent with `GPT-5.5` achieves the highest overall *VSTR*, whereas Claude Code with `GLM-5.2` achieves the largest overall Relative Gain (24.52%). On ReAct-OT, nearly 99% *VSTR* still yields limited Relative Gain, reflecting the difficulty of the task. These results show that the scientific tasks remain challenging, while our environments reliably reveal differences in how agents translate valid experimentation into scientific progress.

5.3 Does the Feedback Remain Effective in Long-Term Exploration?

Experimental Settings. To understand how the proposed feedback design shapes long-term exploration, we conduct 24-hour autonomous research runs on eight challenging tasks with clear room for further improvement: *NanoGPT Speedrun*[2], *CELLECT* (Zhou et al., 2025) and six tasks from *MLE-bench* (Chan et al., 2025). We use SciEnBuilder to convert each task into a scientific environment with a reproduced baseline, task-specific evaluator, and protected feedback interface. We retain the native evaluation metric and optimization direction of each task. For example, for MLE-bench tasks, we use the corresponding Kaggle leaderboard metric and additionally record whether the achieved

[2] `https://github.com/KellerJordan/modded-nanogpt`

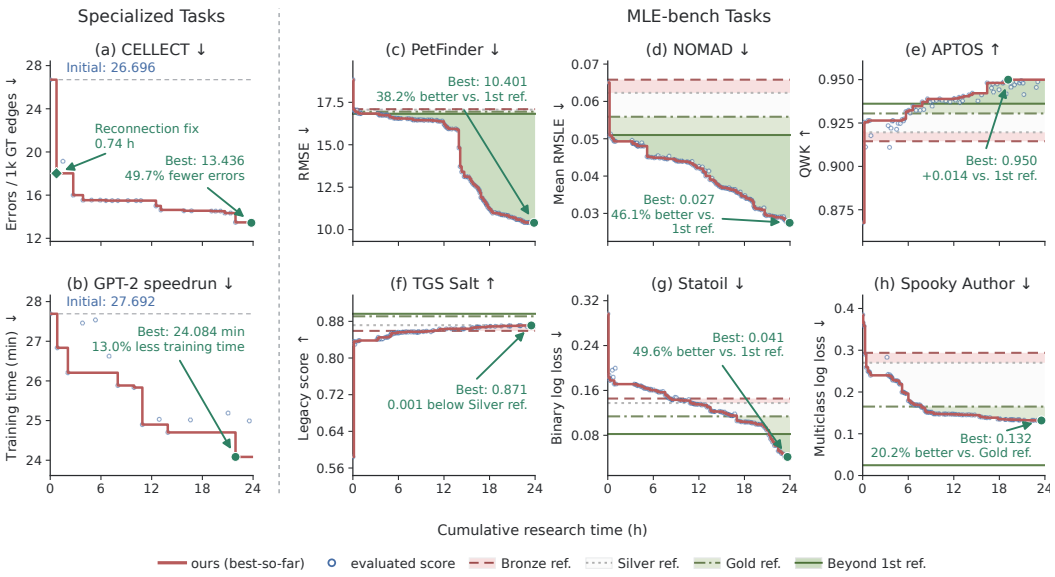


Figure 5: **Optimization trajectories of `gpt-6-astra` through the Codex harness for eight scientific tasks over 24 hours of exploration.** Green annotations highlight key milestones and final results. For the MLE-bench tasks, horizontal background bands mark score ranges corresponding to the historical medal tiers and the first-place reference.

score reaches a medal threshold. Long-term exploration is then conducted with `gpt-6-astra` through the Codex harness for 24 hours on each task, without an explicit limit on API cost.

The eight 24-hour trajectories (Figure 5) show that trustworthy feedback enables coding agents to sustain long-horizon scientific exploration, achieving consistent improvements across diverse scientific tasks and surpassing strong historical reference results.

Progress emerges through discrete breakthroughs. Agent progress often consisted of long exploration periods without measurable gains, followed by discrete breakthroughs that achieved new best results. For example, in CELLECT, five candidates showed no improvement before C14 reduced the error rate after 5.93 additional hours. These patterns suggest that feedback helped agents identify useful directions among unsuccessful attempts and progressively refine their hypotheses. While the evaluation signal remained stable, agent analyses remained adaptive, with median character 3–5-gram TF–IDF similarity ranging from 0.35 to 0.53 across tasks. Together, stable evaluation and adaptive reasoning enabled cumulative progress.

Failed trials provide useful research signals. A representative example is Spooky Author. After iteration 73 improved the score by introducing an XL source, iteration 74 showed that the XL source was complementary rather than a replacement for the existing large6 source, as removing large6 degraded performance on short texts and the HPL class. The agent incorporated both components in the next iteration and achieved further improvement. Similar patterns emerged throughout the trajectories, where failed experiments provided evidence for refining subsequent decisions. Each trial helped identify which hypotheses or interventions should be retained or discarded. We further discuss the limitations of our work in Appendix A.

6 Conclusion

In this work, we studied how environmental feedback shaped the scientific exploration of coding agents. Through controlled experiments, we showed that feedback design fundamentally influenced agent behavior and that unreliable signals could lead to shortcut exploitation, objective mismatch, and unstable exploration. We then introduced SciEnBuilder, an automated framework for constructing trustworthy scientific environments from paper–repository pairs. Experiments on real-world tasks demonstrated that these environments improved research validity, eliminated observed shortcut behaviors, and supported sustained scientific exploration. These results highlighted the importance of trustworthy environments as a foundation for autonomous scientific discovery.

AI USE STATEMENT

Generative AI tools were used as coding agents in our experiments, as described in the experimental setup, and were also used to assist with language editing and improving the readability of the manuscript. All research questions, experimental designs, evaluation protocols, results, and scientific claims were determined and verified by the authors. All AI-assisted text was reviewed and revised by the authors, and all experimental outputs were validated through the evaluation procedures described in the paper. The authors take full responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Increasingly autonomous scientific agents raise concerns about unreliable experimentation and unintended behavior when operating without trustworthy evaluation and appropriate oversight. Our work directly studies this issue by examining how environmental feedback shapes agent behavior (Section 3) and by constructing isolated scientific environments with protected evaluators and controlled feedback interfaces (Section 4). Our environments are constructed from publicly available research papers and code repositories (Table 8) and do not involve sensitive personal data. Human experts manually constructed and cross-checked the controlled environment variants in Section 3 and assessed the quality and faithfulness of the environments generated by SciEnBuilder in Section 5.1. The proposed framework is intended to support reliable scientific exploration, and human oversight remains important when applying such systems to high-risk domains or real-world settings.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our experiments throughout the paper and appendix. The implementation details of **SciEnBuilder** are described in Section 4. Experimental settings, including the coding-agent configurations, feedback conditions, and evaluation metrics are provided in Sections 3.2, 5 and Appendix B. The full task list is displayed in Table 8. The human evaluation details are included in Appendix B. Code and resources for reproducing the environments and experiments will be released upon publication.

REFERENCES

Carlo Abate and Filippo Maria Bianchi. Maxcutpool: differentiable feature-aware maxcut for pooling in graph neural networks. In *International Conference on Learning Representations*, volume 2025, pp. 52506–52533, 2025.

Yonas Atinafu and Robin Cohen. Rewardhackingagents: Benchmarking evaluation integrity for llm ml-engineering agents. In *2026 IEEE 42nd International Conference on Data Engineering Workshops (ICDEW)*, pp. 27–34. IEEE, 2026.

Oren Avram, Berkin Durmus, Nadav Rakocz, Giulia Corradetti, Ulzee An, Muneeswar G Nittala, Prerit Terway, Akos Rudas, Zeyuan Johnson Chen, Yu Wakatsuki, et al. Accurate prediction of disease-risk factors from volumetric medical scans by a deep vision model pre-trained with 2d scans. *Nature biomedical engineering*, 9(4):507–520, 2025.

Vahid Balazadeh Meresht, Hamidreza Kamkari, Valentin Thomas, Junwei Ma, Bingru Li, Jesse Cresswell, and Rahul Krishnan. Causalpfn: Amortized causal effect estimation via in-context learning. *Advances in Neural Information Processing Systems*, 38:154945–154984, 2026.

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *International Conference on Learning Representations*, volume 2024, pp. 24814–24836, 2024.

Tadesse Destaw Belay, Ahmed Haj Ahmed, Alvin Grissom II, Iqra Ameer, Grigori Sidorov, Olga Kolesnikova, and Seid Muhie Yimam. Culemo: Cultural lenses on emotion-benchmarking llms for cross-cultural emotion understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18894–18909, 2025.

Yoshua Bengio, Michael Cohen, Damiano Fornasiere, Joumana Ghosn, Pietro Greiner, Matt MacDermott, Sören Mindermann, Adam Oberman, Jesse Richardson, Oliver Richardson, et al. Superintelligent agents pose catastrophic risks: Can scientist ai offer a safer path? *arXiv preprint arXiv:2502.15657*, 2025.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, volume 2025, pp. 96934–96990, 2025.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Chenru Duan, Guan-Horng Liu, Yuanqi Du, Tianrong Chen, Qiyuan Zhao, Haojun Jia, Carla P Gomes, Evangelos A Theodorou, and Heather J Kulik. Optimal transport for generating transition states in chemical reactions. *Nature Machine Intelligence*, 7(4):615–626, 2025.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, et al. Towards general agentic intelligence via environment scaling. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17610–17621, 2026.

Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. Researchgym: Evaluating language model agents on real-world ai research. *arXiv preprint arXiv:2602.15112*, 2026.

Manuel Gloeckler, Michael Deistler, Christian Dietrich Weilbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=DL79HYCFFq.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*, volume 2025, pp. 77899–77935, 2025.

William Timothy Gowers, Ben Green, Freddie Manners, and Terence Tao. On a conjecture of marton. *Annals of Mathematics*, 201(2):515–549, 2025.

Panagiotis D Grontas, Antonio Terpin, Efe Balta, Raffaello D'Andrea, and John Lygeros. Pinet: Optimizing hard-constrained neural networks with orthogonal projection layers. In *International Conference on Learning Representations*, volume 2026, pp. 62933–62979, 2026.

Xiao Han, Saima Absar, Lu Zhang, and Shuhan Yuan. Root cause analysis of anomalies in multivariate time series through granger causal discovery. In *International Conference on Learning Representations*, volume 2025, pp. 45137–45153, 2025.

Jialiang Huang, Hongxuan Tang, Jingchang Chen, Yuxuan Liu, Yixiao Chen, Yuan Cheng, Yi Tao, Jingli Zhou, Yupeng Chen, Haoyu Chen, Jiarui Wang, Shengkai Lin, Chuqi Zhang, Bryan Lee Teng, Lian Guo, Zhe Fu, Wenjun Gao, Yisong Wang, Liang Zhao, Zehao Wang, Ziwei Xie, Yongqiang Guo, Peixin Cong, Ziyi Gao, Shuiping Yu, Hanwei Xu, Zuofan Wu, Zhizhou Ren, Yuyang Zhou, Bowei Zhang, Zhihuan Huang, Qihao Zhu, Lei Wang, Tianle Lin, Han Yu, Jiewen Hu, Dejian Yang, Shuo Yang, Shanghao Lu, Shaoyuan Chen, Junjie Qiu, Zhangli Sha, Yinmin Zhong, Yongtong Wu, Shiyu Wang, Wei Liu, Bingzheng Xu, Longhao Chen, Qiushi Du, Yuzhen Huang, Shirong Ma, Yaohui Wang, Mingshu Chen, Tongrui Xiong, Y. C. Yan, Haowen Luo, Haofen Liang, Xiaokang Zhang, Weihao Zeng, Runxin Xu, Peiyi Wang, Jinhua Zhu, Ruoyu Zhang, Wenkai Yang, Qi Tang, Jiping Yu, Tian Ye, Ruizhe Pan, Honghui Ding, Xiaodong Liu, Lingxiao Luo, Zhihong Shao, Yuhan Wu, Jibai Lu, Wen Liu, Haoling Zhang, Jingcheng Hu, Yaoyang Ye, Chaofan Lin, Zhaochen Zhang, Jianan Tong, Hengxu Wu, Zhihao

Li, Yicheng Wang, Luyao Wang, Yuzhuo Bai, Lingyue Fu, Ruifan Xu, Y. Z. Wang, Zonglin Li, Mingqi Wei, Haiyang Shen, Chengyuan Zhang, Chao Jin, Zili Zhang, R. H. Yang, Xinbo Xu, Jian Zhou, Ruidong Zhu, Yuzhe Guo, Zelun Pan, Shaoheng Nie, Erhang Li, Shuhan Lin, Zheng Liu, Anshuo Chen, Zilong Lyu, Sinuo Cao, Rui Yu, Chuhao Wang, Junyi Guo, Junxiao Song, Kaifeng Chen, Menghao Ye, Junxian Li, Di Wu, Haiyang Ma, Yilun Wang, Haoran Yang, Yizai Cai, Shichun Liu, Yiping Wang, Junbo Sun, Shicheng Xu, Xiao Bi, Ying He, Yichao Zhang, Mingxing Zhang, Liyue Zhang, Panpan Huang, and Wenfeng Liang. Deepseek elastic compute (dsec): A sandbox infrastructure for effective agentic training at scale, 2026. URL https://arxiv.org/abs/2609.22978.

Jiajie Jin, Yuyang Hu, Kai Qiu, Qi Dai, Chong Luo, Guanting Dong, Xiaoxi Li, Tong Zhao, Xiaolong Ma, Gongrui Zhang, Zhirong Wu, Bei Liu, Zhengyuan Yang, Linjie Li, Lijuan Wang, Hongjin Qian, Yutao Zhu, and Zhicheng Dou. Toward generalist autonomous research via hypothesis-tree refinement. *arXiv preprint arXiv:2606.11926*, 2026.

Xurui Li, Ziming Huang, Feng Xue, and Yu Zhou. Musc: Zero-shot industrial anomaly classification and segmentation with mutual scoring of the unlabeled images. In *International Conference on Learning Representations*, volume 2024, pp. 2172–2197, 2024.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26 (226):1–16, 2025.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, volume 2025, pp. 33132–33156, 2025.

Cristiano Migali, Gianmarco Genalti, Alberto Maria Metelli, and Marco Mussi. Learning to rank from incomplete rankings. In *Forty-third International Conference on Machine Learning*, 2026.

Hanane Nour Moussa, Yifei Li, Zhuoyang Li, Yankai Yang, Cheng Tang, Tianshu Zhang, Nesreen K Ahmed, Ali Payani, Ziru Chen, and Huan Sun. D3-gym: Constructing real-world verifiable environments for data-driven discovery. *arXiv preprint arXiv:2604.27977*, 2026a.

Hanane Nour Moussa, Yifei Li, Zhuoyang Li, Yankai Yang, Cheng Tang, Tianshu Zhang, Nesreen K. Ahmed, Ali Payani, Ziru Chen, and Huan Sun. D3-gym: Constructing real-world verifiable environments for data-driven discovery. In *The Second Workshop on Language Models for Scientific Discovery*, 2026b. URL https://openreview.net/forum?id=e6yoh28VUu.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Junhyuk Oh, Gregory Farquhar, Iurii Kemaev, Dan A Calian, Matteo Hessel, Luisa Zintgraf, Satinder Singh, Hado Van Hasselt, and David Silver. Discovering state-of-the-art reinforcement learning algorithms. *Nature*, 648(8093):312–319, 2025.

Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24838–24848. IEEE, 2025.

Jesús Pineda, Sergi Masó-Orriols, Montse Masoliver, Joan Bertran, Mattias Goksör, Giovanni Volpe, and Carlo Manzo. Enhanced spatial clustering of single-molecule localizations with graph neural networks. *Nature Communications*, 16(1):9693, 2025.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar VK, Rongzhi Zhang, Changhao Li, Ian Wong, Sherry Yang, Percy Liang, Chao Zhang, et al. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering. *Advances in Neural Information Processing Systems*, 38, 2026.

Martiño Ríos-García, Nawaf Alampara, Chandan Gupta, Indrajeet Mandal, Sajid Mannan, Ali Asghar Aghajani, NM Krishnan, and Kevin Maik Jablonka. Ai scientists produce results without reasoning scientifically. *arXiv preprint arXiv:2604.18805*, 2026.

Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning. In *International Conference on Learning Representations*, volume 2026, pp. 38867–38932, 2026.

Zachary S Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=BsMMc4MEGS.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

Igor Sterner and Simone Teufel. Minimal pair-based evaluation of code-switching. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18575–18598, 2025.

Haotian Sun, Yuchen Zhuang, Wei Wei, Chao Zhang, and Bo Dai. BBox-adapter: Lightweight adapting for black-box large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=jdRIaUu3xY.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

Liyan Tang, Philippe Laban, and Greg Durrett. Minicheck: Efficient fact-checking of llms on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8818–8847, 2024.

Binghai Wang, Chenlong Zhang, Dayiheng Liu, Jiajun Zhang, Jiawei Chen, Mingze Li, Mouxiang Chen, Rongyao Fang, Siyuan Zhang, Xuwu Wang, et al. The verification horizon: No silver bullet for coding agent rewards. *arXiv preprint arXiv:2606.26300*, 2026a.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

Ren Wang, Kevin Frans, Pieter Abbeel, Sergey Levine, and Alexei Efros. Prioritized generative replay. In *International Conference on Learning Representations*, volume 2025, pp. 47151–47169, 2025.

Yubang Wang, Chenxi Zhang, Bowen Chen, Zezheng Huai, Zihao Dai, Xinchi Chen, Yuxin Wang, Yining Zheng, Jingjing Gong, and Xipeng Qiu. Researchenvbench: Benchmarking agents on environment synthesis for research code execution. *arXiv preprint arXiv:2603.06739*, 2026b.

Yuru Wang, Lejun Cheng, Yuxin Zuo, Sihang Zeng, Bingxiang He, Che Jiang, Junlin Yang, Yuchong Wang, Kaikai Zhao, Weifeng Huang, et al. Naturebench: Can coding agents match the published sota of nature-family papers? *arXiv preprint arXiv:2606.24530*, 2026c.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. Deepscientist: Advancing frontier-pushing scientific findings progressively. In *International Conference on Learning Representations*, volume 2026, pp. 47981–48037, 2026.

Junxi Wu, Dongjian Hu, Yajie Bao, Shu-Tao Xia, and Changliang Zou. Error-quantified conformal inference for time series. In *International Conference on Learning Representations*, volume 2025, pp. 81566–81593, 2025a.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. In *International Conference on Learning Representations*, volume 2025, pp. 58337–58353, 2025b.

Qiujie Xie, Qiming Feng, Tianqi Zhang, Qingqiu Li, Linyi Yang, Yuejie Zhang, Rui Feng, Liang He, Shang Gao, and Yue Zhang. Human simulacra: Benchmarking the personification of large language models. In *International Conference on Learning Representations*, volume 2025, pp. 11169–11211, 2025a.

Qiujie Xie, Yixuan Weng, Minjun Zhu, Fuchen Shen, Shulin Huang, Zhen Lin, Jiahui Zhou, Zilan Mao, Zijie Yang, Linyi Yang, et al. How far are ai scientists from changing the world? *arXiv preprint arXiv:2507.23276*, 2025b.

Amy Xin, Jiening Siow, Junjie Wang, Zijun Yao, Fanjin Zhang, Jian Song, Lei Hou, and Juanzi Li. Eurekagent: Agent environment engineering is all you need for autonomous scientific discovery. *arXiv preprint arXiv:2606.13662*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zihan Yu, Jingtao Ding, and Yong Li. Discovering network dynamics with neural symbolic regression. *Nature Computational Science*, 6(2):156–168, 2026.

Fuwei Zhang, Xiaoyu Liu, Xinyu Jia, Yingfei Zhang, Shuai Zhang, Xiang Li, Fuzhen Zhuang, Wei Lin, and Zhao Zhang. Multi-level relevance document identifier learning for generative retrieval. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10066–10080, 2025.

Xuan Zhang, Jacob Helwig, Yuchao Lin, Yaochen Xie, Cong Fu, Stephan Wojtowytsch, and Shuiwang Ji. Sinenet: Learning temporal dynamics in time-dependent partial differential equations. In *International Conference on Learning Representations*, volume 2024, pp. 11612–11653, 2024.

Tong Zheng, Xidong Wu, Zheng Zhang, Zhankui He, Chaoyi Zhang, Benjamin Coleman, Ruoqiao Wei, Di Bai, Haolin Liu, Rui Liu, Xue Wang, Yue Zhuan, Wang-Cheng Kang, Renkai Xiang, Heng Huang, Xinwu Cheng, and Yunsong Guo. Dream-RSI: Recursive self-improvement through evolving worlds. *arXiv preprint arXiv:2609.14858*, 2026.

Haowen Zhong, Yilan Liu, Haibin Sun, Yuru Liu, Rentao Zhang, Baochen Li, Yi Yang, Yuqing Huang, Fei Yang, Frankie S Mak, et al. Towards global reaction feasibility and robustness prediction with high throughput data and bayesian deep learning. *Nature Communications*, 16(1):4522, 2025.

Hongyu Zhou, Seonghoon Kim, Zhifeng Zhao, Jiaqi Fan, Wen Huang, Xinghua Sui, Lizhi Shao, Haoran An, Jing-Ren Zhang, Jiamin Wu, et al. Cellect: contrastive embedding learning for large-scale efficient cell tracking. *Nature Methods*, 22(11):2411–2422, 2025.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Jiayi Liu, Xiangjun Fan, Zhuokai Zhao, and Hong Yan. Synthetic sandbox for training ML engineering agents. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=zPfNpMtadi.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, et al. Edgebench: Unveiling scaling laws of learning from real-world environments. *arXiv preprint arXiv:2607.05155*, 2026.

Heming Zou, Yunliang Zang, Wutong Xu, Yao Zhu, and Xiangyang Ji. Flylora: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. *Advances in Neural Information Processing Systems*, 38:10386–10419, 2026.

A Discussion

Limitations. Our study focuses on computational scientific tasks where research progress can be evaluated through executable experiments and quantitative metrics. This setting enables controlled analysis of how environmental feedback shapes long-horizon exploration, but does not fully capture scientific discovery processes involving theoretical reasoning or physical experiments. Moreover, although we evaluate several representative coding agents, future agent architectures may exhibit different behaviors under the same environmental feedback. While our results demonstrate that trustworthy environments support sustained refinement, whether they can facilitate genuinely novel discoveries beyond optimization of existing objectives remains an open challenge.

The position of our work. Our work takes an environment-centric view of automated scientific discovery. While prior efforts primarily improve the capabilities of coding agents, we study the complementary question of how scientific environments shape their research behavior. To our knowledge, this is of how environmental feedback influences long-horizon scientific exploration across different coding agents and feedback conditions. We further argue that reliable autonomous discovery requires a clearer separation of roles: **the agent should make research decisions, while the environment should provide independent and trustworthy evaluation.** Allowing the same agent to both decide what to explore and determine whether the result is successful can blur the distinction between scientific progress and self-justified optimization. Trustworthy environments should instead preserve the scientific objective, and provide aligned feedback throughout iterative exploration. SciEnBuilder serves as a concrete realization of this perspective.

The Cost. Our analysis involves **392** full research trajectories, totaling **1085** agent-hours, **1,426** research iterations, and over $20,000 in API costs. Each run is allocated up to 4-24 hours, with an average duration of 2.93 h across 392 recorded runs (Figure 19). Many trajectories terminate early as agents adaptively decide when to stop exploration, indicating that practical cost depends on both the allocated budget and exploration behavior. Importantly, long-duration runs do not always reflect productive exploration. Three Original-environment trajectories spent most of their budget on setup and compatibility issues without completing a valid experiment, highlighting the importance of preconfigured environments in converting wall-clock budgets into effective research time.

Reward hacking as an environment-design failure. Our results suggest that reward hacking can arise from the interaction between an agent and the feedback exposed by its environment. We observe two distinct failure modes. In *C0*, exposing the ground-truth evaluator effectively makes the evaluation process part of the optimization loop. Although *C0* achieves the strongest *True* performance across all three tasks, extensive audited shortcut behavior shows that these gains can come from exploiting the evaluator rather than advancing the scientific objective. In *C2*, by contrast, we observe no audited shortcuts and a high *VSTR* of 89.32%, yet the proxy score improves in 33/36 matched comparisons while the True score decreases in 31/36. Here, the agent continues to conduct valid experiments, but optimizes a misspecified target. The instability in *C3* further indicates that hiding evaluators or exposing multiple scores does not resolve the problem when the feedback signals encode competing objectives.

These results show that problematic behavior can emerge simply from optimizing the signals provided by the environment, without assuming deceptive intent. This motivates our designs that isolate protected evaluators, ground scores in executed artifacts and align feedback with the scientific claim. Across the comparison in Section 5.2, SciEnBuilder increases *I-VRCR* from 45.86% to 80.95% and *VSTR* from 76.91% to 94.14%, while reducing audited shortcut behaviors from 206 to 0. Our design mitigates reward hacking to a substantial extent, while stronger agents may still discover new shortcuts beyond the coverage of the current tasks and audits, motivating continued adversarial testing and validation (Wang et al., 2026a; Atinafu & Cohen, 2026).

B Experimental Details

B.1 Motivation Experiments

Environment construction. For each task, we build C0 from the code repository released with the original paper, preparing the data, model assets, and dependencies needed to run the starting method. C0 gives the agent access to this executable repository together with the task's ground-truth

evaluator $f^\star$ and the reference labels used for evaluation. To construct C1–C3, we use the same task workspace, remove existing scoring entry points, evaluation answers, cached scores, and scored reference outputs, and then add the evaluation resources specified by each condition. Task code, permitted data, execution outputs, and format checks remain available. Across C0–C3, we keep the scientific objective, starting method, model assets, resource budget, and final evaluation rule fixed. Each environment is cross-checked by a second expert (Section 3.2).

Following Section 3.1, C1 omits official performance evaluation while retaining ordinary execution feedback f_{t+1}. The agent π_θ can develop its own surrogate objective from the available materials. C2 supplies a proxy evaluator $\tilde{f}$, with parameters selected to produce candidate rankings that diverge from $f^\star$ while keeping scores within a usable range. C3 exposes $\mathrm{API}(f^\star, \tilde{f})$, returning two scores for each candidate while keeping both evaluator implementations and protected labels on the host.

Human Simulacra. We use a fixed `Mistral-7B-Instruct-v0.1` model, the MACM method, and a fixed set of 11 characters and 165 questions. The proxy combines response features such as answer format, text length, option density, and variation across repeated generations. Its parameters are selected offline using true scores. During agent runs, it operates on response features alone.

CELLECT. The workspace contains the released checkpoint, labeled S3 training data, and S1 development images. C1 withholds S1 labels, C2 and C3 provide feedback on S1, and C0 additionally exposes S2 evaluation resources. The proxy combines localization, local association, motion, and lineage-topology terms, with weights selected for divergent candidate preferences.

MIRO. We use the Scenario 8 checkpoint, three labeled training fields of view (FOVs), and 47 evaluation FOVs. C2 and C3 provide feedback on the training FOVs; C0 also exposes evaluation labels. The proxy favors clustering granularity subject to coverage and precision constraints. Point identifiers are replaced and row order is shuffled in every condition to remove label information encoded in these fields.

Evaluation metrics. *True Score and Lazy Score.* For the final candidate artifact x_r in run r, we report $f^\star(x_r)$ and $\tilde{f}(x_r)$ on the task's final evaluation data. Both are computed after the run under a common protocol across C0–C3. Human Simulacra uses the released-source scoring rule, normalized to 0–100 and averaged over three generation seeds, on the frozen prediction bundle. CELLECT reports lineage errors per 1,000 ground-truth edges on S2 (lower is better). MIRO reports the unweighted mean $\mathrm{ARI}^\dagger$ over the 47 evaluation FOVs (higher is better). For CELLECT and MIRO, host-side replay generates the predictions scored by both evaluators. Lazy Scores follow the corresponding task's optimization direction.

Valid Scientific Time Rate (VSTR). VSTR combines scientific workflow occupancy with a C1-specific measure of objective consistency. Let T_r be the total duration of active agent sessions, S_r the duration attributed to scientific work, and D_r the duration of scientific time segments flagged by audited shortcut actions. The shortcut-adjusted occupancy is

$$W_r = \frac{\max(0, S_r - D_r)}{T_r}, \qquad T_r > 0. \tag{3}$$

Active sessions are identified from native agent logs. Scientific work includes task and code inspection, method implementation, experiment execution, evaluation, and result analysis. Background-job waiting and reasoning between actions are included when linked to scientific work through timestamped records. Unattributed gaps remain in T_r and are excluded from S_r. Overlapping sessions and concurrent activities are consolidated before measuring duration. Each scientific segment with positive-duration overlap with a retained shortcut action contributes its full duration to D_r once.

In C1, Q_r measures agreement between the agent's surrogate objective or recorded candidate choices and $f^\star$. We align score directions and exclude pairs tied under either ordering. For M_r comparable pairs, of which K_r agree, we compute

$$Q_r = \max\left(0, \frac{2K_r}{M_r} - 1\right), \qquad \mathrm{VSTR}_r = \begin{cases} W_r Q_r, & \text{in } C_1, \\ W_r, & \text{in } C_0, C_2, C_3. \end{cases} \tag{4}$$

The transformation uses 50% pairwise agreement as its chance baseline and clips below-chance agreement to zero. For $M_r = 0$, Q_r and C1 VSTR are undefined.

In C0, C2, and C3, VSTR measures shortcut-adjusted scientific workflow occupancy. In C1, it also accounts for agreement with the true objective, combining time allocation with objective consistency. Sustained work on a misaligned proxy can therefore yield high VSTR and poor True performance in C2. We report VSTR as a percentage and average the run-level values arithmetically.

Shortcut Behavior Number. We count audited shortcut behaviors involving evaluation scores, reference answers, or protected scoring resources. The audit tracks C0 use of exposed true-evaluation feedback and reference labels, access to protected resource identities, and direct shortcut findings in the run logs. C2 and C3 feedback queries through the prescribed interface are treated as routine evaluation.

Manual review removes false positives from both shortcut counts and time deductions. Confirmed shortcuts without action timestamps contribute to the count alone.

Full results. Figures 6, 7, and 8 show C3 candidate performance in recording order, centered on the initial candidate. Tables 4, 5, and 6 report each task–agent–condition combination over three repeats: mean and median True/Lazy Scores, mean VSTR, and summed Shortcut counts. Table 1 averages scores and VSTR over 12 runs per task and condition, and sums Shortcuts over the 36 runs per condition. The counts for C1, C2, and C3 are zero after manual review. Relative to C1, C2 improves mean Lazy performance in all 12 task–agent configurations and worsens mean True performance in 11. Claude Code + `GLM-5.2` on Human Simulacra improves both means. C3 improves mean True performance over C2 in all 12 configurations.

Table 4: Per-agent results for Human Simulacra over three repeated runs. VSTR is averaged and Shortcut counts are summed across repeats. Best mean scores and VSTR within each condition are bolded at the displayed precision.

Environment	Harness	Model	True Score ↑		Lazy Score ↑		VSTR ↑ (%)	Shortcuts ↓
			Mean	Median	Mean	Median		
C0	Codex	GPT-5.5 high	47.39	47.27	37.64	37.70	79.30	5
	RG-Agent	GPT-5.5 high	**62.22**	57.70	28.92	29.58	**98.88**	10
	Claude Code	GLM-5.2 high	48.32	48.97	**40.47**	42.91	82.50	18
	Codex	GLM-5.2 high	50.22	50.06	36.55	35.20	83.15	11
C1	Codex	GPT-5.5 high	46.71	47.03	37.05	36.70	59.28	0
	RG-Agent	GPT-5.5 high	**59.64**	61.82	30.14	30.38	**68.04**	0
	Claude Code	GLM-5.2 high	43.31	44.97	**40.48**	38.30	22.13	0
	Codex	GLM-5.2 high	42.95	45.33	31.70	31.95	22.46	0
C2	Codex	GPT-5.5 high	27.39	27.15	51.73	54.38	98.29	0
	RG-Agent	GPT-5.5 high	42.59	48.85	45.80	31.47	98.40	0
	Claude Code	GLM-5.2 high	**44.32**	44.85	44.19	44.54	**99.70**	0
	Codex	GLM-5.2 high	12.40	11.27	**71.21**	75.25	98.34	0
C3	Codex	GPT-5.5 high	44.77	44.97	**44.93**	44.82	97.03	0
	RG-Agent	GPT-5.5 high	**63.52**	60.85	40.83	46.12	**99.27**	0
	Claude Code	GLM-5.2 high	47.03	46.18	36.39	33.18	94.60	0
	Codex	GLM-5.2 high	45.17	44.97	42.69	41.15	98.23	0

Table 5: Per-agent results for CELLECT over three repeated runs. VSTR is averaged and Shortcut counts are summed across repeats. Best mean scores and VSTR within each condition are bolded at the displayed precision.

Environment	Harness	Model	True Score ↓		Lazy Score ↓		VSTR ↑ (%)	Shortcuts ↓
			Mean	Median	Mean	Median		
C0	Codex	GPT-5.5 high	**19.87**	17.73	**859.69**	859.42	65.87	17
	RG-Agent	GPT-5.5 high	25.53	28.07	861.92	863.83	7.16	7
	Claude Code	GLM-5.2 high	22.17	20.68	860.43	859.44	**90.43**	6
	Codex	GLM-5.2 high	19.95	17.14	877.95	863.83	53.58	24
C1	Codex	GPT-5.5 high	28.24	28.22	862.51	861.30	31.24	0
	RG-Agent	GPT-5.5 high	29.43	24.44	867.80	867.33	62.50	0
	Claude Code	GLM-5.2 high	22.88	21.43	**861.38**	863.16	88.09	0
	Codex	GLM-5.2 high	**21.24**	22.02	866.89	865.59	**98.74**	0
C2	Codex	GPT-5.5 high	877.96	923.50	400.90	258.88	97.80	0
	RG-Agent	GPT-5.5 high	615.88	894.04	481.81	325.47	82.40	0
	Claude Code	GLM-5.2 high	**326.47**	28.07	662.30	863.83	98.26	0
	Codex	GLM-5.2 high	927.07	928.86	**250.32**	253.79	**98.88**	0
C3	Codex	GPT-5.5 high	24.12	20.80	858.98	859.20	97.81	0
	RG-Agent	GPT-5.5 high	23.20	20.78	860.46	859.31	73.05	0
	Claude Code	GLM-5.2 high	**19.44**	20.68	**858.34**	858.05	97.91	0
	Codex	GLM-5.2 high	21.83	20.80	861.85	863.03	**98.72**	0

Table 6: Per-agent results for MIRO over three repeated runs. VSTR is averaged and Shortcut counts are summed across repeats. Best mean scores and VSTR within each condition are bolded at the displayed precision.

Environment	Harness	Model	True Score ↑		Lazy Score ↑		VSTR ↑ (%)	Shortcuts ↓
			Mean	Median	Mean	Median		
C0	Codex	GPT-5.5 high	0.89	0.88	**0.95**	0.99	77.02	3
	RG-Agent	GPT-5.5 high	**0.94**	0.91	0.92	0.88	7.06	3
	Claude Code	GLM-5.2 high	0.92	0.92	0.72	0.72	**87.02**	9
	Codex	GLM-5.2 high	0.93	0.92	0.68	0.70	77.35	8
C1	Codex	GPT-5.5 high	0.83	0.81	**0.91**	0.91	42.44	0
	RG-Agent	GPT-5.5 high	0.75	0.79	0.74	0.83	31.50	0
	Claude Code	GLM-5.2 high	**0.88**	0.90	0.84	0.87	64.86	0
	Codex	GLM-5.2 high	0.87	0.89	0.90	0.89	**91.72**	0
C2	Codex	GPT-5.5 high	0.16	0.00	**1.00**	1.00	85.78	0
	RG-Agent	GPT-5.5 high	0.20	0.00	**1.00**	1.00	**91.23**	0
	Claude Code	GLM-5.2 high	**0.73**	0.83	0.97	0.99	90.55	0
	Codex	GLM-5.2 high	0.29	0.33	**1.00**	1.00	89.74	0
C3	Codex	GPT-5.5 high	0.83	0.83	0.92	0.92	80.07	0
	RG-Agent	GPT-5.5 high	**0.87**	0.88	**0.96**	0.99	58.63	0
	Claude Code	GLM-5.2 high	**0.87**	0.89	0.91	0.91	**93.55**	0
	Codex	GLM-5.2 high	0.85	0.89	0.90	0.89	91.87	0

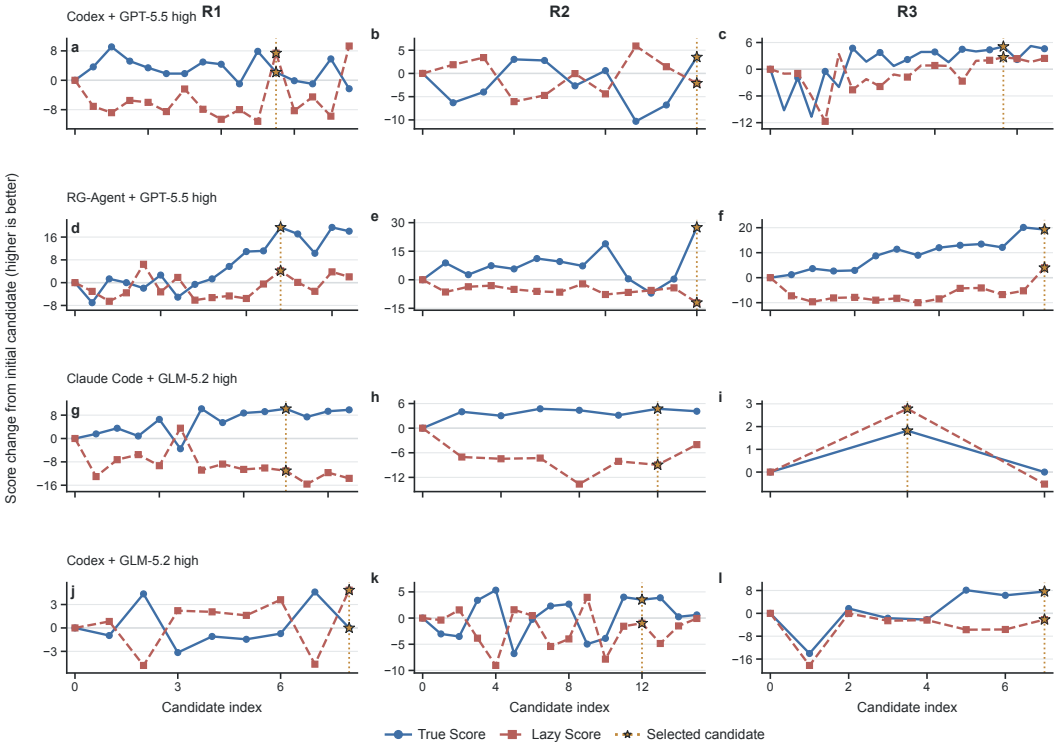


Figure 6: **Candidate performance under C3 on Human Simulacra.** Rows are agent configurations and columns are repeats. Blue and red curves show True and Lazy Score changes from the initial candidate, computed from frozen prediction bundles on the fixed question set. The horizontal axis follows candidate recording order. Stars mark candidates used for the reported final scores.

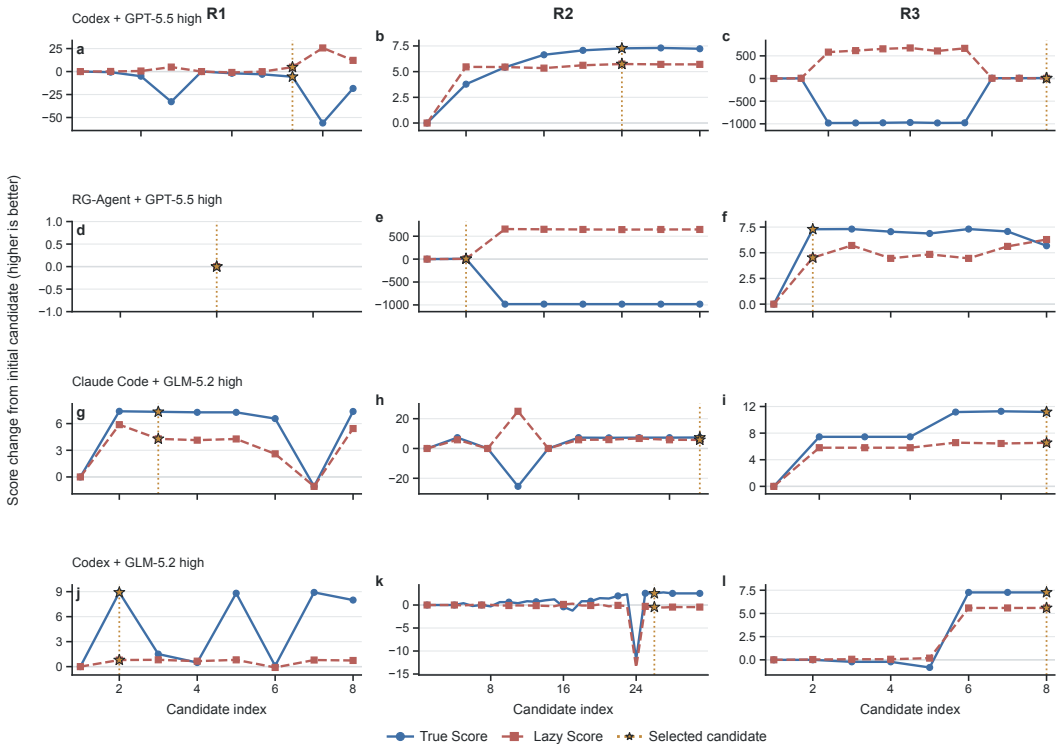


Figure 7: **Candidate performance under C3 on CELLECT.** Rows are agent configurations and columns are repeats. Curves show post-run True and Lazy Score changes on S2 in candidate recording order, sign-reversed so that higher is better. Online feedback uses S1. Stars mark final-score candidates; RG-Agent R1 contains a single accepted baseline candidate.

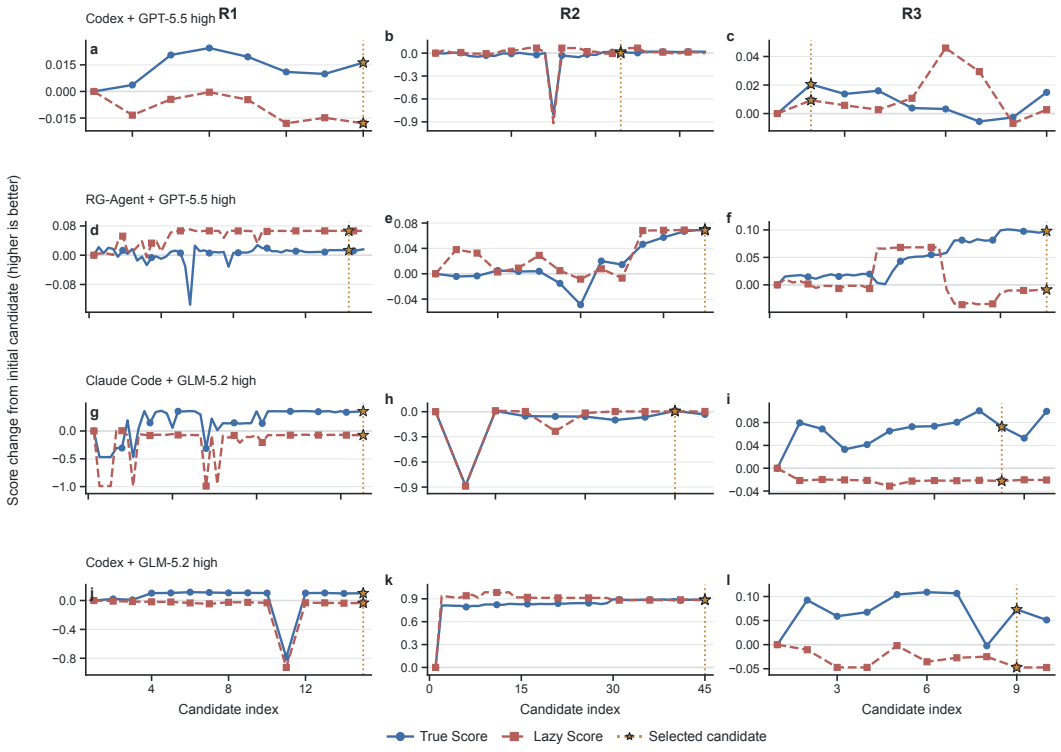


Figure 8: **Candidate performance under C3 on MIRO.** Rows are agent configurations and columns are repeats. Curves show post-run True and Lazy Score changes on the 47 evaluation FOVs in candidate recording order. Online feedback uses three training FOVs. Stars mark final-score candidates, including the host-recovered RG-Agent R1 and R2 outputs described in the text.

Prompt

```
You are an autonomous research agent working inside the provided
repository workspace.

Your goal is to improve the target scientific result under the stated
resource constraints. Inspect the repository and available assets,
reproduce the current baseline when possible, identify the
bottleneck, implement and evaluate a justified improvement, and use
the feedback available in the environment to decide the next action.

Before finishing, save all required artifacts and provide a concise
report containing: the task pursued, commands executed, changes made,
measured result, and artifact locations.
```

Figure 9: Prompt for coding agents.

B.2 Main Results

Comparison Setup. We compare our constructed environments with the original repositories under the same evaluation protocol and coding-agent configurations. For the original repositories, we report only aggregate results because frequent shortcut exploitation compromises task-level scientific-score comparisons.

Evaluation Metrics. We evaluate the validity of the experimental process and the resulting scientific progress. **I-VRCR** is the micro-averaged proportion of valid research iterations across runs. **Shortcuts** counts audited behaviors that bypass the intended experimental process. **VSTR** measures the proportion of execution time devoted to valid scientific activities, and **Start Time** is the elapsed time before the first successful experiment. Feedback effectiveness is measured by $\mathrm{FIR}_{\mathrm{Macro}}$, the average run-level feedback improvement rate, and $\mathrm{FIR}_{\mathrm{Pooled}}$, which pools improved feedback transitions across all runs. **Relative Gain** measures progress from the task-specific baseline toward its optimum, accounting for whether the task objective is maximized or minimized. Higher I-VRCR, VSTR, FIR, and Relative Gain are better, whereas fewer shortcuts and shorter start times are preferred.

B.3 Human Evaluation

We assessed the constructed scientific environments through a two-stage human audit. Reviewers inspected the paper, task, environment, grader, claim tree, logs, and other artifacts relevant to their assigned criteria.

Evaluation process. Three human reviewers independently audited all 30 environments constructed by SciEnBuilder for *Environment Executability*, *Grader Faithfulness*, and *Claim Tree Faithfulness*. We then sampled 10 of the 30 environments for evaluation by two additional experts. These experts independently assessed *Environment Executability*, *Grader Faithfulness*, *Claim Tree Faithfulness*, *Claim Tree Completeness*, and *Task Clarity*. All reviewers submitted their assessments before seeing or discussing others' judgments.

Judgments and issue severity. Each criterion received **Pass** or **Concern**. Pass required all core checks to be satisfied, sufficient supporting evidence, and no Critical or Major issues; all other cases received Concern. Minor issues were recorded and could accompany a Pass. For each Concern, reviewers specified whether it reflected an observed problem or insufficient evidence.

Reviewers rated each issue by its impact, independently of repair effort. A **Critical** issue could lead to an incorrect scientific conclusion, acceptance of an invalid submission, exposure of hidden answers, or failure of the core task. A **Major** issue affected a core claim, primary metric, key evidence, or main workflow. A **Minor** issue reduced clarity, completeness, or reproducibility without changing the core conclusion or blocking the main workflow.

Evaluation criteria. Table 7 summarizes the five criteria.

Table 7: Human evaluation criteria.

Metric	Core criterion
Grader Faithfulness	The grader faithfully computes the primary metric reported in the original paper.
Claim Tree Completeness	The claim tree includes all key claims and supporting evidence from the original paper within the task scope.
Claim Tree Faithfulness	The claims and evidence relations in the claim tree accurately reflect the original paper.
Environment Executability	The documentation enables a user to understand how to operate and verify the environment.
Task Clarity	The task clearly explains what the user is expected to accomplish.

Audit procedure. Reviewers followed the steps below in order.

1. **Paper reference.** Before inspecting the claim tree or grader, reviewers recorded the core scientific question, up to three main claims, the primary metric and its direction, the dataset and split, the key baseline and setting, and the paper's reference result. Unreported or ambiguous information was flagged and left unresolved.
2. **Environment execution.** Reviewers followed the official documentation to attempt setup, baseline execution, artifact location, and grading. They recorded commands, exit codes, manual interventions, timeouts, logs, and output locations. After a failure, they preserved the original record before making at most one minimal diagnostic intervention, without changing official code or artifacts merely to make the run succeed.
3. **Result traceability.** For at least one core result, or all core results if there were three or fewer, reviewers verified a single-run chain: reported result $\rightarrow$ command and configuration $\rightarrow$ execution log $\rightarrow$ artifact $\rightarrow$ grader output. The chain required matching metric names and values from a single run. A broken chain for a core result was normally Major, or Critical if it could support a false high score or scientific conclusion.
4. **Grader audit.** Reviewers checked the metric definition, data, split, sample set, processing, aggregation, and threshold. They also checked that the grader scored the artifact directly, rejected missing required fields, and kept hidden answers private. They independently recomputed the primary metric once and compared it with the grader output under the prespecified tolerance.
5. **Artifact tests.** Before testing, reviewers specified the expected outcome for: (A) a correct artifact from the real baseline, which should pass; (B) a schema-valid artifact with one controlled, score-relevant error, which should fail or score substantially lower; and (C) a cheating or invalid artifact, which should be rejected or fail. If artifact B passed, the issue was rated at least Major. Awarding a false high score to artifact C was Critical.
6. **Stability.** Reviewers ran the grader five times on the same fixed artifact and recorded every metric value and final judgment. Deterministic graders had to return identical results; stochastic graders had to remain within the prespecified tolerance. Inconsistent Pass/Fail judgments were at least Major.
7. **Claim tree and task.** Reviewers compared the claim tree with the frozen paper reference, recording omissions separately from unfaithful statements or links. They then assessed whether the task instructions alone gave a first-time user the required task and submission format, with no need to consult the paper, source code, or oral guidance.
8. **Final assessment.** For each criterion, reviewers recorded a Pass or Concern judgment, supporting evidence, and relevant issue IDs. Issue records specified the affected object, severity, evidence, likely effect, and suggested fix. A summary of at most 100 words identified the most reliable component, the most serious problem, and any unresolved evidence gap or ambiguity.

C SCIENBUILDER

Section 3 shows that coding agents organize their research around the feedback they receive, usually a score computed from their final output. Interpreting this score as scientific evidence requires checking that the executed experiment matches the paper's claim and protocol (Huang et al., 2026). SciEnBuilder explicitly links the claim, execution, and score. We formalize this connection in Section C.1 and give an overview of SciEnBuilder in Section C.2. The remaining sections describe how it fixes the scientific intent (Section C.3), certifies evaluation (Section C.4), and returns feedback (Section C.5).

C.1 PROBLEM FORMULATION

Research target. A paper contains multiple interdependent claims, while its repository provides executable evidence for only a subset of them. We represent these claims as a *research DAG* $\mathcal{G}$ whose nodes are typed as data, method, evaluation protocol, or result. Its edges record how evidence flows: a method is *evaluated by* a protocol, which *supports* a result under stated data *conditions*. Different claims rely on different datasets, protocols, and experimental budgets. We therefore define each environment around one *research target*:

$$\tau = (\sigma,\ p,\ \Theta,\ \kappa), \tag{5}$$

where $\sigma \subseteq \mathcal{G}$ is the evidence path to be tested; p is its evaluation protocol, comprising splits, reference, metric and direction, aggregation, baseline x_0, and a tolerance ϵ_p for nondeterminism; Θ is the candidate space, i.e., the changes the agent may make while everything else is held fixed; and κ records claims outside the executable scope and conclusions that cannot be supported by the environment. For $x \in \Theta$, let $f_p^\star(x)$ be the outcome of executing p faithfully on x. This is the ground-truth evaluator $f^\star$ of Section 3.1, now tied to a specific claim. Since $f_p^\star$ is not directly observable, the environment approximates it with a score $g(x)$.

Research validity. Environment-construction and paper-to-code pipelines are usually judged by *reproduction*: the baseline runs and recovers the reported number, i.e., $|g(x_0) - f_p^\star(x_0)| \le \epsilon_p$ (Seo et al., 2026; Starace et al., 2025; Wang et al., 2026b). A research agent repeatedly evaluates new candidates beyond x_0 and, as Section 3.3 shows, tends to explore regions where the score can be improved most easily. Let $\mathcal{X}_{\pi_\theta}(g) \subseteq \Theta$ be the candidates that agent π_θ can reach while interacting with g. A research environment requires

$$|g(x) - f_p^\star(x)| \le \epsilon_p \qquad \forall\, x \in \mathcal{X}_{\pi_\theta}(g). \tag{6}$$

Reproduction verifies this condition at x_0, while research requires it to hold throughout the candidates explored by the agent.

Checkable conditions. The set $\mathcal{X}_{\pi_\theta}(g)$ is revealed only as research proceeds, so Eq. equation 6 cannot be verified in advance. We therefore identify the main ways in which g may deviate from $f_p^\star$, and introduce a checkable condition for each:

- **Faithful**: p matches the protocol used by the paper to support the claims on σ, including its metric, split, and aggregation.
- **Grounded**: $g(x)$ is recomputed from artifacts produced by executing x.
- **Isolated**: no $x \in \Theta$ can read or modify the reference, the scorer, or the records on which g depends.
- **Anchored**: $g(x_0)$ recovers the paper-reported result within ϵ_p.

Faithful aligns g with $f_p^\star$, preventing the proxy optimization observed in C2. Grounded and Isolated ensure that changes in $g(x)$ come from the executed artifacts, preventing the evaluator exploitation observed in C0. Anchored checks that the baseline reproduces the paper-reported result within the allowed tolerance. Faithful and Anchored are verified during construction; Grounded and Isolated are checked for each evaluated candidate. These checks verify Eq. equation 6 on every candidate explored by the agent. The feedback design in Section C.5 addresses the presentation issues behind drift in C1 and objective switching in C3.

Assurance level. Each environment is released at the level certified from its available artifacts: BLOCKED $\prec$ EXECUTION $\prec$ VALIDATION $\prec$ TRUSTED. Missing checkpoints or metrics that can

be verified but not recomputed may limit the attainable level. At EXECUTION, the baseline runs; VALIDATION additionally requires its output artifacts to pass protocol checks; TRUSTED further requires Grounded and Isolated scores under a frozen contract. Anchored is reported separately because reproduction gaps may reflect released checkpoints, hardware, or other factors. A release below the declared level requires explicit human approval.

C.2 SciEnBuilder Overview

SciEnBuilder follows this split between construction and research (Figure 3). Construction freezes a research target as an experimental contract (Section C.3), then builds, evaluates, and repairs an implementation of it (Section C.4). The result is a *public task package*, containing the agent's workspace, candidate interface, and development data, and a *private evaluator package*, containing the reference and scorer, held on the host. During research, each evaluation is bound to the contract and returned with structured evidence (Section C.5).

Coding agents interpret papers, adapt code, and diagnose failures. Execution produces evidence, and deterministic code verifies all resulting artifacts and decisions, manages identifiers, source anchors, and digests, and determines acceptance. This separates agent proposals from verification. Each object (claim, protocol, run, artifact, score, or repair) carries a content digest and links to related objects by identifier.

C.3 Fixing the Scientific Intent as a Contract

The connection between a repository's computations and the paper's claims is often implicit. Renamed metrics, hard-coded splits, and unreleased scripts behind reported results can lead to selecting an executable metric that differs from the paper's intended evaluation. SciEnBuilder first examines the paper and code separately to determine what the environment should measure.

Isolated audits. A *paper audit* extracts claims, protocols, and reported results, each with its source location. A *code audit* inventories executable entry points, data loaders, metrics, and scorers. Conducted independently of the paper audit, it provides an unbiased account of the repository's implementation. Reconciliation produces claim–protocol bindings that record whether the repository implements the full protocol or its execution alone. In the latter case, the metric is built from the paper or taken from an official external scorer. Unmatched items record evidence gaps between the paper's protocol and the implementation.

Research DAG and target. The reconciled claims form $\mathcal{G}$. Each claim carries a category, a statement, a falsification condition, and source anchors, and result nodes also record the paper-reported observations. The builder selects an evidence path σ that is scientifically central and executable within budget, and instantiates p with concrete splits, baseline, metric, and aggregation. The selected target may cover only a subset of the paper while preserving the original experimental semantics. Experiments outside the executable scope are moved to κ.

Freezing. The target τ is serialized as an *experimental contract* and frozen by a content digest before execution is planned. All subsequent stages reference this digest, and repairs change the implementation while preserving the target. Each protocol field carries a source anchor so readers can check its faithfulness against the paper.

C.4 Isolated Realization, Grounded Evaluation, and Repair

With the contract fixed, we implement the execution conditions:

Isolated realization. The contract is compiled into two packages. The public package contains code, public development splits, and an interface for submitting a frozen candidate. The private package holds the hidden splits, reference, and scorer behind a host-side evaluation service. Before release, service credentials are removed from the agent's environment, labels are stripped from every task-side data view, and the task package is scanned for residual reference content. Candidate submission is the sole route to obtaining a score.

Grounded evaluation. The builder runs the baseline end to end through the same scoring path used for agent candidates. A score is Grounded when (i) it is recomputed from execution artifacts, (ii) a

manifest binds the scored artifacts to their producing run and the target's split, protocol, scorer, and reference, and (iii) the reference has verified authority. The manifest lets readers trace each score to its run, protocol, and reference. Manifests mixing artifacts across splits, runs, protocols, or references are rejected. SciEnBuilder reuses the authors' evaluation code when available to preserve the original metric definition. Comparing the baseline score with the paper-reported result determines Anchored.

Contract-preserving repair. SciEnBuilder repairs dependencies, resources, and evaluators within the frozen experimental contract. Each failure is assigned a matching write scope: runtime errors go to *repository* repair, missing data or models to *resource* repair, and scoring failures to *evaluator* repair. When execution succeeds and only scoring fails, the outputs are reused and the scorer is repaired, avoiding an expensive rerun. Every repair is validated against the contract before being committed. The contract and reference remain outside all repair scopes; repairs requiring changes to either are referred to a human. Affected stages are re-executed and revalidated after each repair.

C.5 Feedback as Evidence

The same scalar outcome can reflect scientific failure, execution failure, incomplete evaluation, or a valid negative result. Let $\mathcal{V}(\mathcal{E})$ denote the official evaluation interfaces available to the agent. SciEnBuilder provides one such interface, returning structured evidence alongside a single optimization score:

$$\mathcal{V}(\mathcal{E}) = \big\{\mathrm{API}(g,\ d)\big\}. \tag{7}$$

Candidates are ranked solely by g, maintaining a single optimization objective across iterations (C2, C3). Diagnostics d from the same run under p report performance by group, error composition, gains and losses relative to the previous best, and the gap between the planned and executed change. The diagnostics exclude instance-level labels. Each result carries an *evidence state* with separate fields for execution (completed, failed, or not run), measurement (valid, invalid, or unavailable), and protocol completion (complete or partial). Its *scope* identifies the relevant claims on σ and the unsupported conclusions in κ.

One research round. Round t proceeds in a fixed order:

1. Before seeing any result, the agent hashes a *precommitment* pre_t that states the change, the hypothesis, the expected metric change, and a falsifier.
2. It freezes and submits the candidate x_t.
3. The host evaluates x_t and issues a receipt r_t that binds the digests of x_t, pre_t, the protocol, the reference, and the evaluator.
4. The agent receives $f_t = \big(g(x_t),\ d(x_t),\ r_t\big)$ with its evidence state and scope.
5. It writes a post-grade report post_t that compares the result with pre_t and states the next falsifiable action.

The next round opens only after r_t and post_t are recorded, with authoritative receipts held by the host. Submissions can outlive their connections; each carries a fixed request identity so retries after uncertain failures resume the original request with the same deadline and budget. Correction and resubmission are permitted only for submissions explicitly rejected before acceptance.

Interpreting score changes. Each result is assessed for performance improvement through g, agreement with the precommitted prediction, and diagnostic support for the proposed mechanism. Each hypothesis receives a deterministic assessment:

$$\rho(\mathrm{pre}_t, f_t) \in \{\textsc{supported},\ \textsc{contradicted},\ \textsc{inconclusive},\ \textsc{not decidable}\}, \tag{8}$$

where a result without a valid measurement is NOT DECIDABLE, and an incomplete protocol permits at most an INCONCLUSIVE assessment. Valid runs that contradict the hypothesis count as completed experiments and contribute negative evidence. A score improvement accompanied by diagnostics inconsistent with the proposed mechanism is recorded as a performance gain with a revised explanation. Receipts bind results to the original predictions, keeping them distinct from post-hoc interpretations.

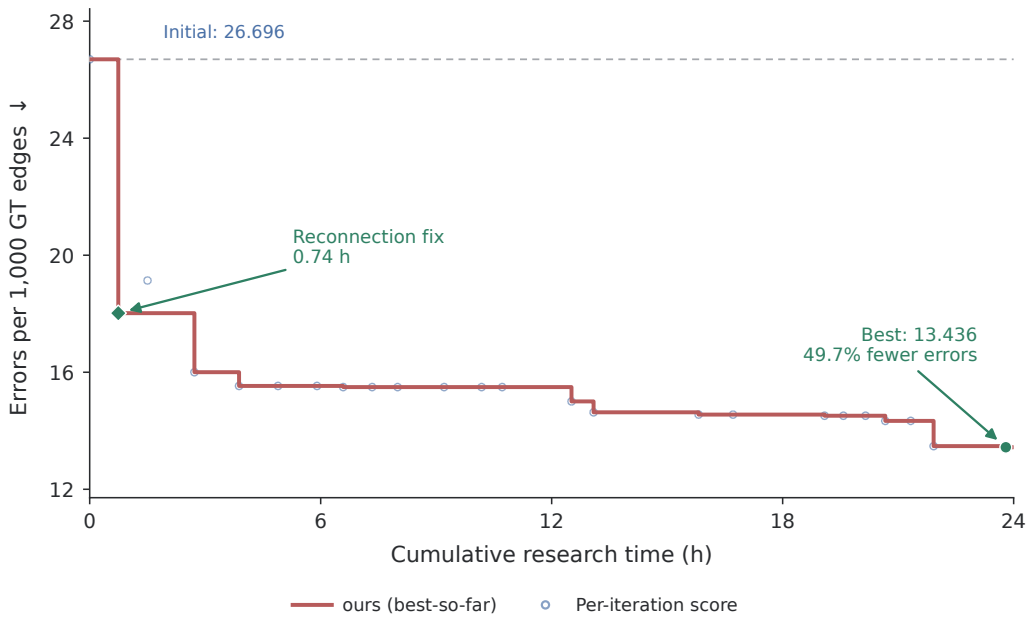


Figure 10: **Optimization trajectory of the CELLECT task over 24 hours.** Blue circles denote evaluated scores, while the red step curve shows the best-so-far trajectory. Lower values indicate better performance. The green annotations highlight the reconnection fix and the final best result.

Findings supported across rounds are appended to a research record with their evidence and scope. The frozen DAG keeps the original paper claims separate from findings produced during agent exploration. Eq. equation 6 establishes measurement validity. Final generalization claims are evaluated on a blind split whenever the protocol provides one unused for feedback.

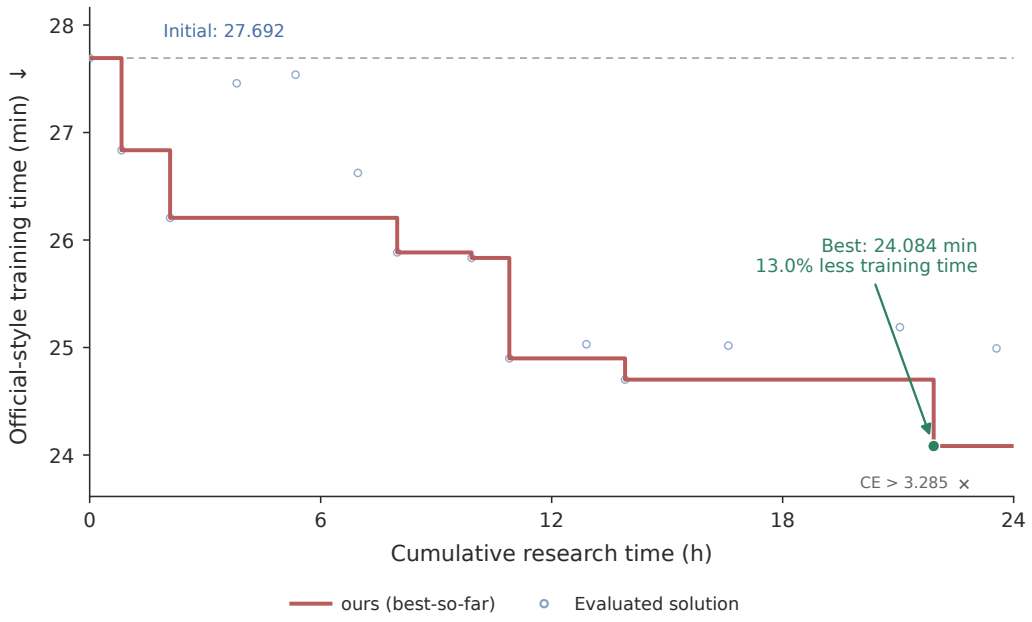


Figure 11: **Optimization trajectory of NanoGPT Speedrun (GPT-2 training) over 24 hours.** Blue circles denote evaluated training times, and the red step curve shows the best-so-far trajectory. Lower training time indicates better performance. Green annotations highlight the first configuration below 25 minutes and the final best result.

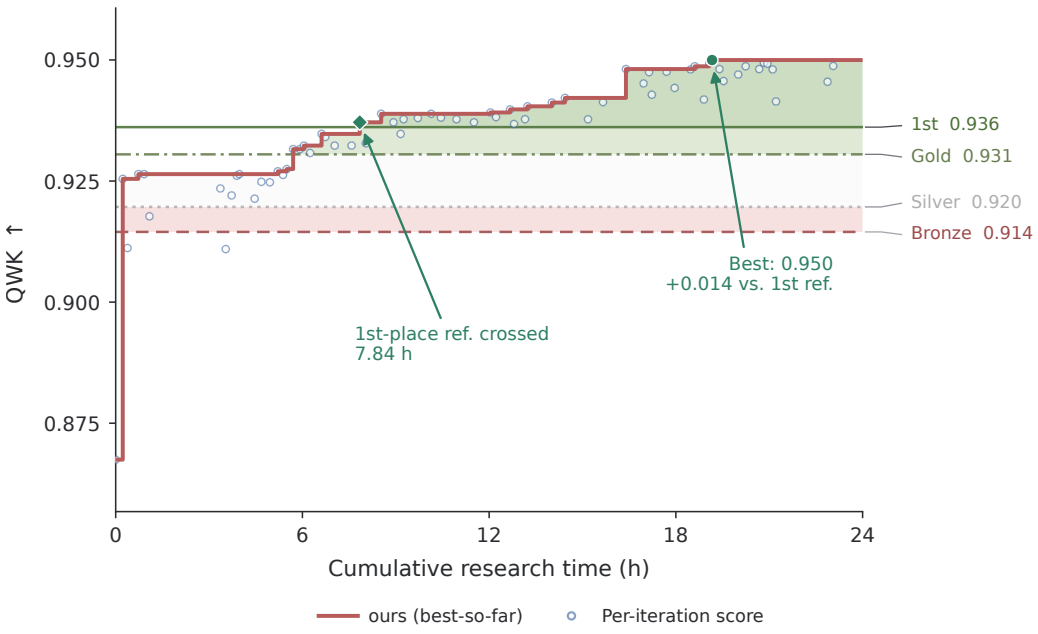


Figure 12: **Optimization trajectory of the APTOS task over 24 hours.** Blue circles denote evaluated QWK values, and the red step curve shows the best-so-far trajectory. Higher QWK is better. Background bands indicate the attained historical reference tiers, while the green annotation highlights the final best score and its improvement over the first-place reference.

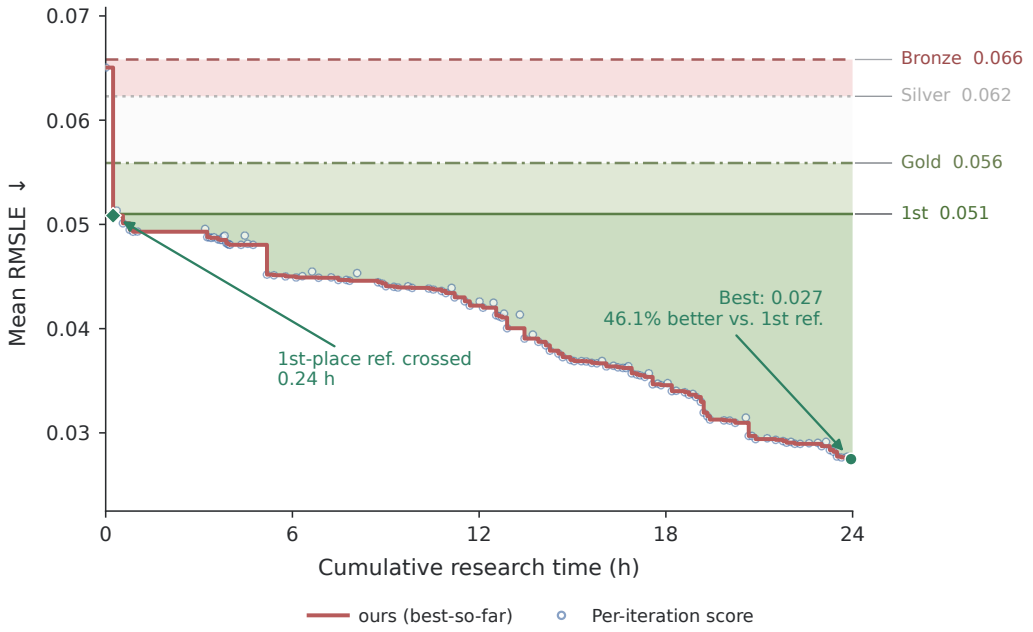


Figure 13: **Optimization trajectory of the NOMAD task over 24 hours.** Blue circles denote evaluated Mean RMSLE values, and the red step curve shows the best-so-far trajectory. Lower values are better. The background bands indicate the attained historical reference tiers, and the green annotation marks the final best result.

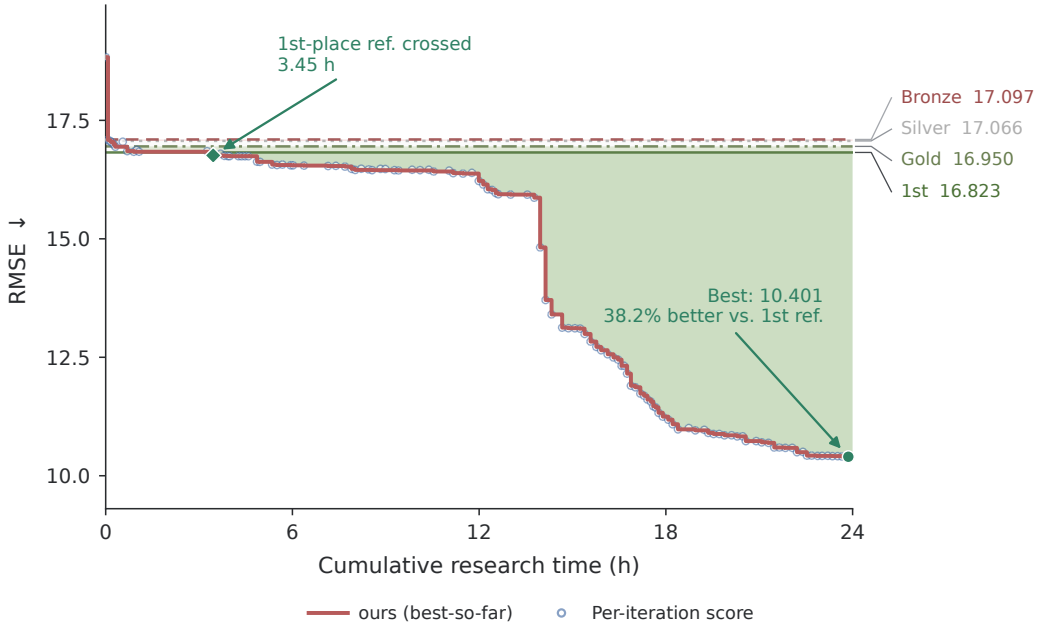


Figure 14: **Optimization trajectory of the PetFinder task over 24 hours.** Blue circles denote evaluated RMSE values, and the red step curve shows the best-so-far trajectory. Lower RMSE is better. Background bands indicate the time intervals during which the incumbent surpassed the bronze, silver, gold, and first-place historical reference levels.

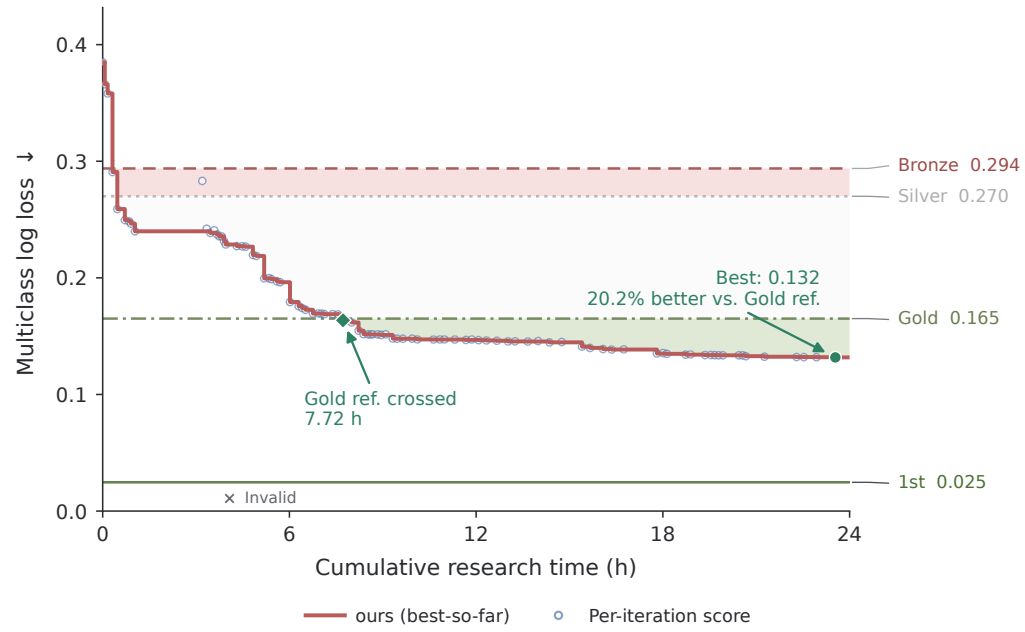


Figure 15: **Optimization trajectory of the Spooky Author task over 24 hours.** Blue circles denote evaluated multiclass log-loss values, and the red step curve shows the best-so-far trajectory. Lower values are better. Background bands indicate the attained historical reference tiers, and the green annotation highlights the final result relative to the gold reference.

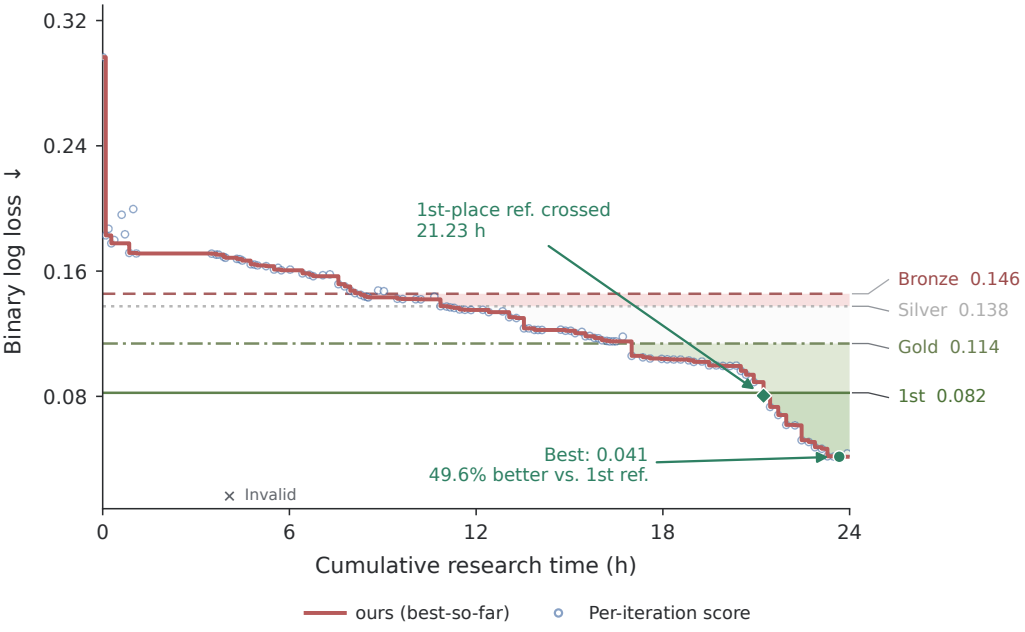


Figure 16: **Optimization trajectory of the Statoil task over 24 hours.** Blue circles denote evaluated binary log-loss values, and the red step curve shows the best-so-far trajectory. Lower values are better. Background bands indicate the attained historical reference tiers, with the final trajectory entering the region beyond the first-place reference.

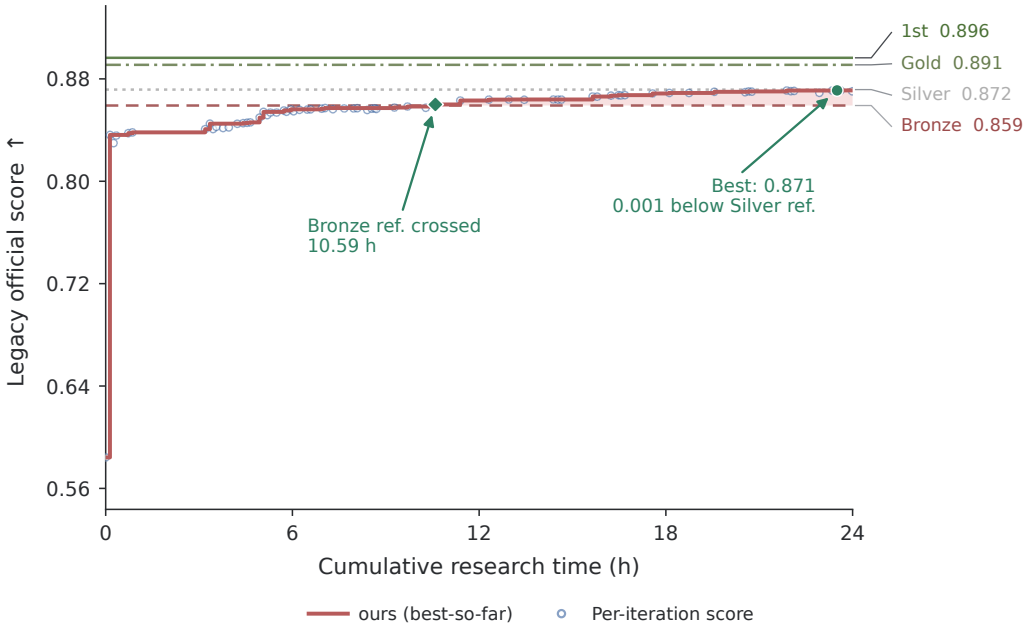


Figure 17: **Optimization trajectory of the TGS Salt task over 24 hours.** Blue circles denote evaluated legacy scores, and the red step curve shows the best-so-far trajectory. Higher values are better. Background bands indicate the attained historical reference tiers, and the green annotation highlights the final best result.

Table 8: Papers and open-source repositories used to construct the scientific environments.

No.	Paper title	Venue	Research area	GitHub repository
1	Human Simulacra: Benchmarking the Personification of Large Language Models (Xie et al., 2025a)	ICLR 2025	Natural Language Processing	hasakiXie123/Human-Simulacra
2	CELLECT: Contrastive Embedding Learning for Large-Scale Efficient Cell Tracking (Zhou et al., 2025)	Nature Methods 2025	Computational Biology	zzz333za/CELLECT
3	Enhanced Spatial Clustering of Single-Molecule Localizations with Graph Neural Networks (Pineda et al., 2025)	Nature Communications 2025	Computational Biology	DeepTrackAI/MIRO
4	Error-Quantified Conformal Inference for Time Series (Wu et al., 2025a)	ICLR 2025	Time-Series Analysis	creator-xi/Error-quantified-Conformal-Inference
5	Minimal Pair-Based Evaluation of Code-Switching (Sterner & Teufel, 2025)	ACL 2025	Natural Language Processing	igorsterner/acs
6	TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling (Gorishniy et al., 2025)	ICLR 2025	Algorithm Optimization	yandex-research/tabm
7	Optimal Transport for Generating Transition States in Chemical Reactions (Duan et al., 2025)	Nature Machine Intelligence 2025	Computational Chemistry	deepprinciple/react-ot
8	MaxCutPool: Differentiable Feature-Aware Maxcut for Pooling in Graph Neural Networks (Abate & Bianchi, 2025)	ICLR 2025	Algorithm Optimization	NGMLGroup/MaxCutPool
9	TALENT: A Tabular Analytics and Learning Toolbox (Liu et al., 2025)	JMLR 2025	Algorithm Optimization	LAMDA-Tabular/TALENT
10	Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature (Bao et al., 2024)	ICLR 2024	Natural Language Processing	baoguangsheng/fast-detect-gpt
11	Root Cause Analysis of Anomalies in Multivariate Time Series through Granger Causal Discovery (Han et al., 2025)	ICLR 2025	Time-Series Analysis	hanxiao0607/AERCA
12	On a Conjecture of Marton (Gowers et al., 2025)	Annals of Mathematics 2025	Computational Mathematics	teorth/pfr
13	All-in-One Simulation-Based Inference (Gloeckler et al., 2024)	ICML 2024	Computational Mathematics	mackelab/simformer
14	OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations (Ouyang et al., 2025)	CVPR 2025	Computer Vision	opendatalab/OmniDocBench
15	Accurate Prediction of Disease-Risk Factors from Volumetric Medical Scans by a Deep Vision Model Pre-Trained with 2D Scans (Avram et al., 2025)	Nature Biomedical Engineering 2025	Computational Biology	cozygene/SLIViT
16	BBox-Adapter: Lightweight Adapting for Black-Box Large Language Models (Sun et al., 2024)	ICML 2024	Algorithm Optimization	haotiansun14/BBox-Adapter

Continued on next page

Table 8 continued

No.	Paper title	Venue	Research area	GitHub repository
17	FlyLoRA: Boosting Task Decoupling and Parameter Efficiency via Implicit Rank-Wise Mixture-of-Experts (Zou et al., 2026)	NeurIPS 2025	Algorithm Optimization	gfyddha/FlyLoRA
18	MuSc: Zero-Shot Industrial Anomaly Classification and Segmentation with Mutual Scoring of the Unlabeled Images (Li et al., 2024)	ICLR 2024	Computer Vision	xrli-U/MuSc
19	Discovering Network Dynamics with Neural Symbolic Regression (Yu et al., 2026)	Nature Computational Science 2026	Network Science	tsinghua-fib-lab/ND2
20	MiniCheck: Efficient Fact-Checking of LLMs on Grounding Documents (Tang et al., 2024)	EMNLP 2024	Natural Language Processing	Liyan06/MiniCheck
21	CULEMO: Cultural Lenses on Emotion — Benchmarking LLMs for Cross-Cultural Emotion Understanding (Belay et al., 2025)	ACL 2025	Natural Language Processing	llm-for-emotion/culemo
22	Multi-Level Relevance Document Identifier Learning for Generative Retrieval (Zhang et al., 2025)	ACL 2025	Information Retrieval	zhangfw123/MERGE
23	ΠNet: Optimizing Hard-Constrained Neural Networks with Orthogonal Projection Layers (Grontas et al., 2026)	ICLR 2026	Algorithm Optimization	antonioterpin/pinet
24	SD-LoRA: Scalable Decoupled Low-Rank Adaptation for Class Incremental Learning (Wu et al., 2025b)	ICLR 2025	Algorithm Optimization	WuYichen-97/SD-Lora-CL
25	Towards Global Reaction Feasibility and Robustness Prediction with High Throughput Data and Bayesian Deep Learning (Zhong et al., 2025)	Nature Communications 2025	Computational Chemistry	Chemlex-AI/bayesian-reactivity-prediction
26	FasterCache: Training-Free Video Diffusion Model Acceleration with High Quality (Lyu et al., 2025)	ICLR 2025	Algorithm Optimization	Vchitect/FasterCache
27	CausalPFN: Amortized Causal Effect Estimation via In-Context Learning (Balazadeh Meresht et al., 2026)	NeurIPS 2025	Causal Inference	vdblm/CausalPFN
28	Learning to Rank from Incomplete Rankings (Migali et al., 2026)	ICML 2026	Information Retrieval	m1gwings/pirate-experiments
29	Prioritized Generative Replay (Wang et al., 2025)	ICLR 2025	Algorithm Optimization	renwang435/pgr
30	SineNet: Learning Temporal Dynamics in Time-Dependent Partial Differential Equations (Zhang et al., 2024)	ICLR 2024	Computational Mathematics	divelab/AIRS

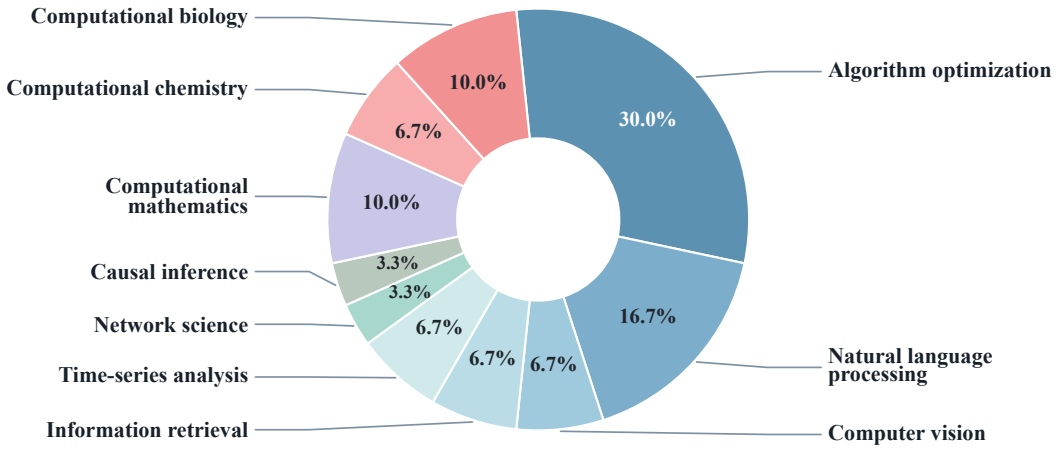


Figure 18: **Distribution of research areas across the 30 scientific environments.** Each environment is assigned to its primary research area. Numbers in parentheses indicate the number of environments, while percentages denote their proportions in the full suite. The suite spans ten research areas, with the largest shares coming from algorithm optimization and natural language processing.

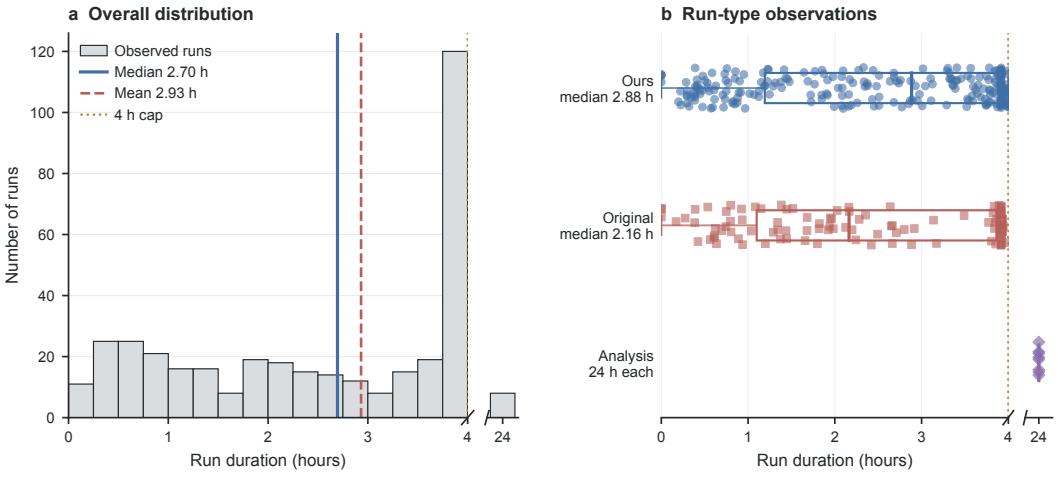


Figure 19: **Distribution of run durations of our experiments.** (a) Overall duration distribution of the runs with recorded durations. The solid and dashed lines indicate the median (2.66 h) and mean (2.46 h), respectively, and the dotted line marks the 4-hour budget limit. (b) Duration distributions under different environments. Each point denotes one run. Boxes indicate the interquartile range and whiskers show the observed range.

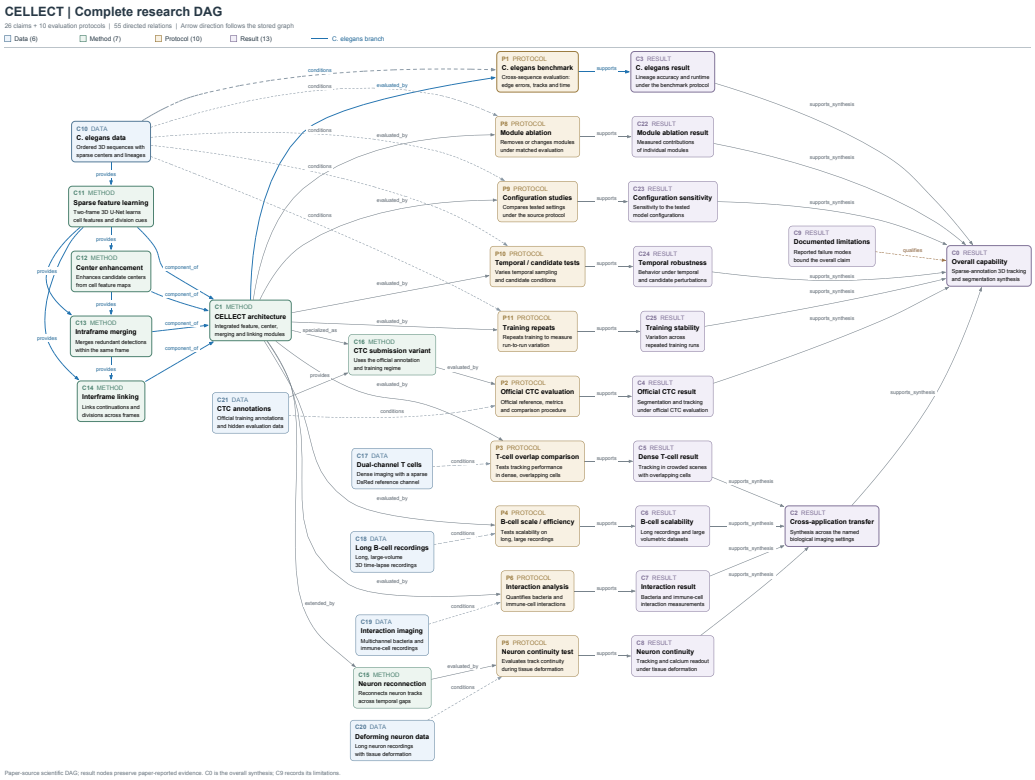


Figure 20: **Audited CELLECT research DAG.** The audited DAG organizes the CELLECT paper into source-traceable claims, methods, protocols, and results. The highlighted path defines a coherent executable research target and provides the basis for constructing $\tau = (\sigma, p, \Theta, \kappa)$.

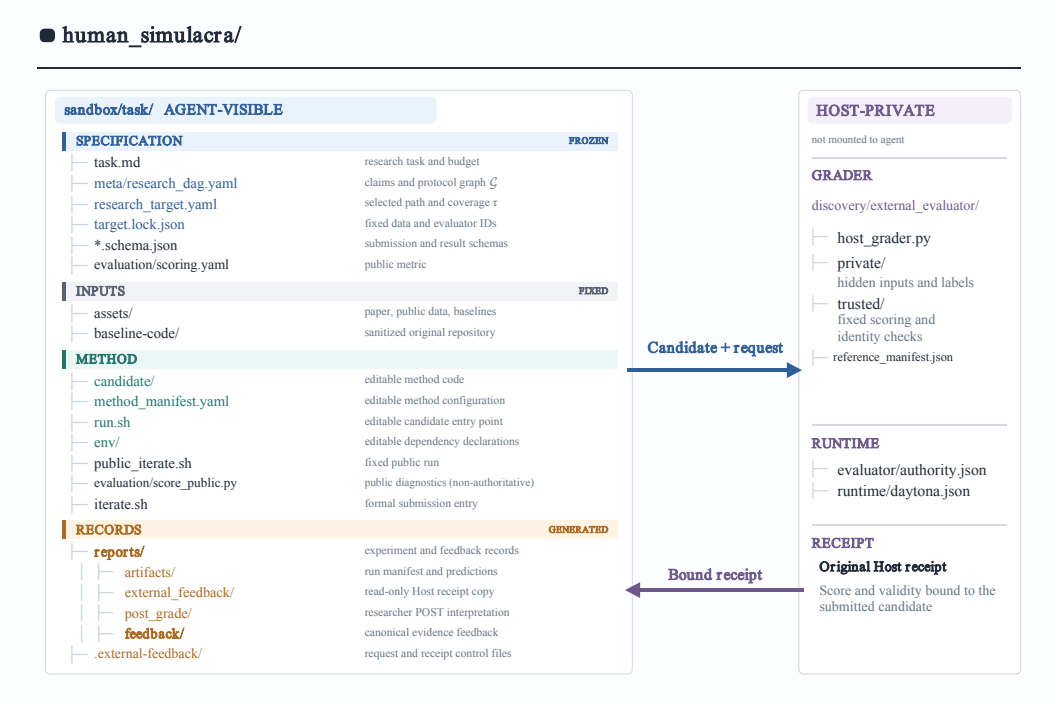


Figure 21: **Human Simulacra sandbox and host boundary.** The agent-visible task package contains the fixed specification, public resources, editable method, and generated research records, while hidden inputs and authoritative evaluation remain Host-private. Candidate submissions are evaluated by the protected grader and returned as bound receipts.

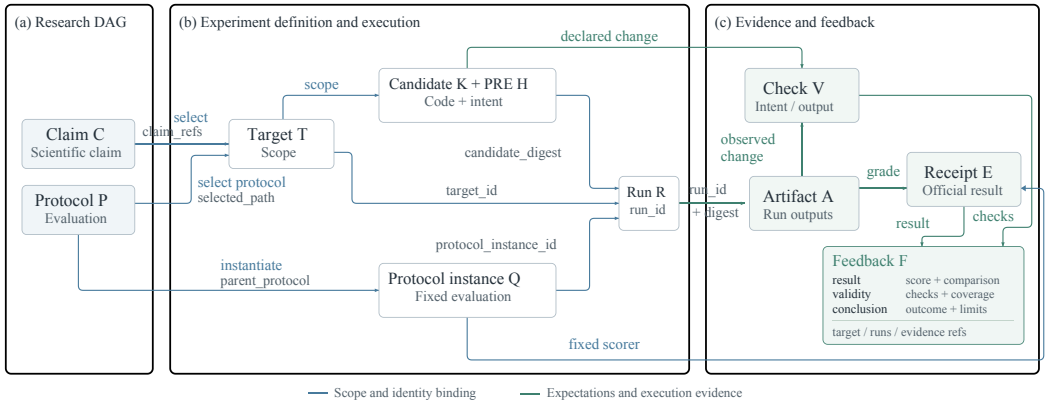


Figure 22: **Scientific intent, execution evidence, and feedback.**The framework links scientific claims and protocols to a scoped target, executable candidate, run artifacts, official evaluation, and structured feedback. Explicit identifiers and evidence bindings preserve traceability from research intent to measured outcomes and conclusions.