ScienceBuddy: Recursive-in-Recursive Self-Improvement for Interactive Scientific Agents

Shuhan Xue[1,*] Jianyuan Zhong[1,*] Ziyuan Nan[1,*] Wenbin Li[1] Zhaochen Yu[1]
Jinchao Ding[1] Qiang Gao[2,3,4,5] Pengyu Zhan[6] Yuntong Zhang[6] Tian Cheng[6]
Zhenfei Yin[1,7,†] Yingcheng Wu[1,8,†] Ling Yang[1,9,†]

Website: Science-Buddy-Product | **Code:** Gen-Verse/ScienceBuddy

Abstract

We introduce and release **ScienceBuddy**, an interactive scientific research workspace that brings continually improving scientific agents into researchers' everyday workflows. ScienceBuddy supports researchers in carrying out scientific tasks while transforming their requests, feedback, and execution evidence into tasks and evaluation rubrics for continual learning. At its core is **recursive-in-recursive self-improvement**, a paradigm that couples harness evolution with model reinforcement learning: the **inner recursion** improves the harness with the model fixed, while the **outer recursion** trains the model under the improved harness. Harness evolution shapes training experience, and model learning creates new opportunities for harness adaptation. We present case studies of researcher interaction, harness refinement, and model learning, with the benchmark cases spanning four scientific task families. By releasing ScienceBuddy as a research product, we make this paradigm available to the scientific community and take a step toward **discovery intelligence**: scientific AI that advances through sustained collaboration with researchers and evolves alongside the research it supports.

Corresponding: yin@phai-labs.com; wuyc@phai-labs.com; yang@phai-labs.com

"Agents will inhabit streams of experience, rather than short snippets of interaction."

— Silver and Sutton, *Welcome to the Era of Experience* (2025) [17, p. 2]

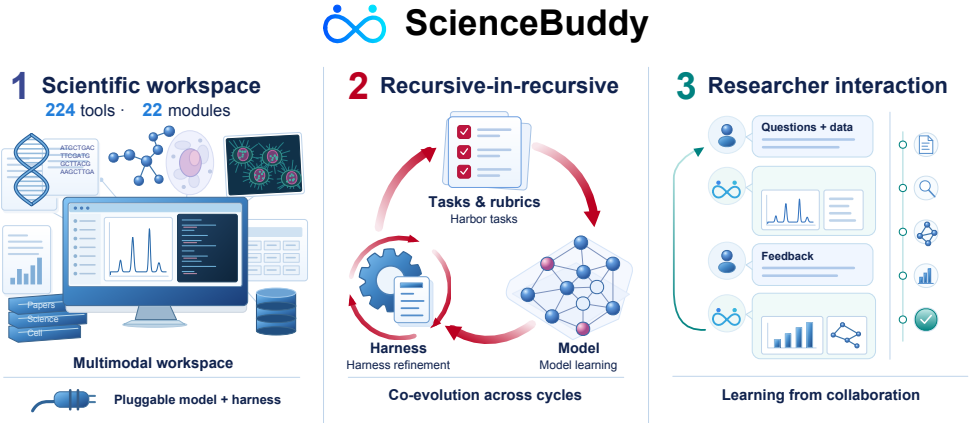

Figure 1 | ScienceBuddy: a scientific workspace that learns through collaboration. Left: Scientific workspace. A multimodal workspace brings together documents, images, tables, and biological sequences with 224 tools across 22 functional modules, spanning genomics, molecular and cancer biology, pharmacology, bioimaging, literature retrieval, and database queries. Pluggable frontier models and agent harnesses support scientific analysis within this shared environment. **Middle: Recursive-in-recursive self-improvement.** Nested harness refinement and model learning are linked through scientific tasks and evaluation rubrics. **Right: Researcher interaction.** Researchers pose questions, inspect results, and refine requirements. These exchanges supply task objectives, evaluation criteria, and evidence for further improvement, connecting scientific collaboration to the next learning cycle.

*Equal contribution.
†Corresponding authors.

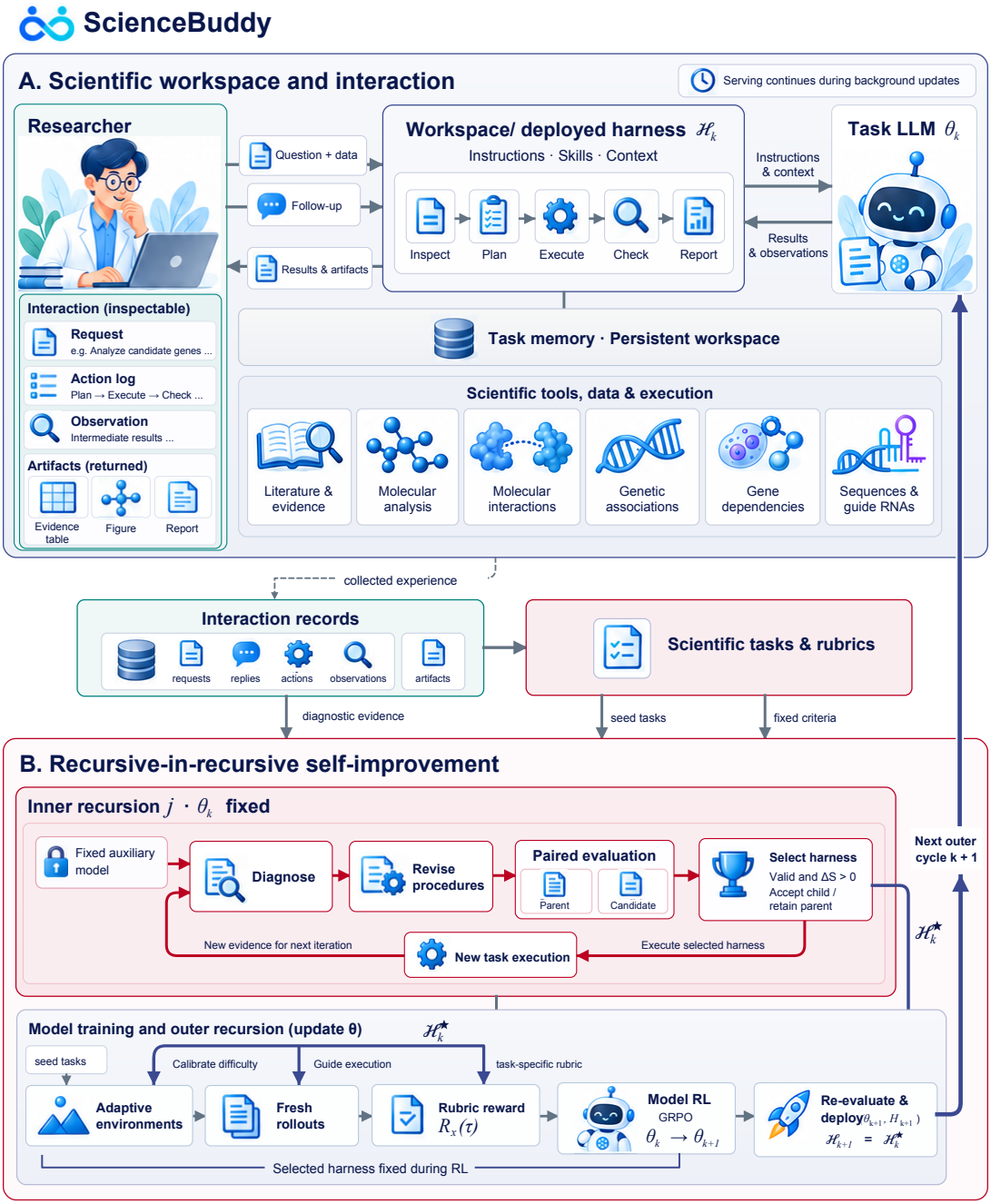

Figure 2 | ScienceBuddy: an interactive scientific workspace with recursive-in-recursive self-improvement. Top: Scientific workspace and researcher interaction. Researchers submit questions and data, inspect execution traces and artifacts, and refine analyses through follow-up exchanges. The deployed harness organizes the task model's instructions, skills, and context, connecting it to scientific tools, data, and a persistent workspace. **Middle: From interactions to learning signals.** Requests, replies, actions, observations, and artifacts provide diagnostic evidence and jointly establish executable scientific tasks and task-specific evaluation rubrics. **Bottom: Coupled harness and model improvement.** In the **inner recursion**, the task model remains fixed while a fixed auxiliary model diagnoses failures and proposes bounded procedural edits. Valid candidates are accepted only when they improve paired development evaluation; execution under the selected harness supplies evidence for further refinement. In the **outer recursion**, the selected harness guides environment-difficulty calibration and fresh on-policy rollouts. Rubric rewards drive GRPO updates to the task model, while the harness and evaluation rubrics remain fixed. The updated model and inherited harness are re-evaluated and deployed together, generating researcher interactions for the next cycle. These background updates proceed while the online service remains available.

Contents

1. Introduction

Scientific research proceeds through analysis, inspection, and revision. Language-model agents can assist by retrieving evidence, querying databases, and executing computational workflows [9, 10, 18]. Researchers then clarify assumptions, question conclusions, and request additional checks. These exchanges reveal how scientific work should be conducted and assessed, but correcting an answer within a conversation does not establish improvement across tasks. This motivates our central question: *How can a scientific agent turn collaboration with researchers into sustained improvements in its working procedures and underlying capabilities?*

Prior work establishes foundations for this problem. Reflection and harness optimization revise reusable instructions and execution procedures [2, 11, 16, 27]; interaction-driven adaptation and rubric-based reinforcement learning provide mechanisms for model improvement [7, 19, 28]. Joint adaptation also has precedent: SIA updates both harnesses and model weights, including for single-cell RNA denoising [8], while HELIX connects harness evolution to model-training data construction [5]. In science, AgentBuild constructs agents from scientist-authored rubrics, curricula, and knowledge bases [15]. We investigate how *collaboration itself* can supply the tasks and assessment criteria that coordinate repeated procedural and policy learning.

We introduce **ScienceBuddy**, an **interactive scientific research workspace for continual learning from researcher collaboration**. Figure 2 provides an overview of the workspace, which combines scientific tools and reference resources [9] with data upload, executable analysis, persistent files, and inspectable traces and artifacts. Researchers refine their requests through dialogue, while a pluggable harness organizes model behavior through instructions, reusable skills, and context-management procedures. Separating this editable harness from the scientific infrastructure makes procedural changes explicit and evaluable. Requests, clarifications, execution records, and artifacts jointly establish task objectives, constraints, and success criteria. We consolidate these criteria into task-specific rubrics and package the corresponding instructions, inputs, and environments as executable Harbor tasks [1]. Researcher replies inform these criteria without serving as unquestioned correctness labels. The resulting tasks support both procedural diagnosis and evaluation of fresh policy rollouts, using executable checks and fixed judges as appropriate.

On this foundation, we propose **recursive-in-recursive self-improvement** (Figure 2). The **inner recursion** holds the task model fixed while a separate, fixed auxiliary model diagnoses failures and proposes bounded edits to instructions, skills, or context settings. Candidates are accepted only when they satisfy edit constraints and improve paired development evaluation [13, 22]. Further execution supplies evidence for the next revision. The **outer recursion** calibrates augmented task environments against the current model and selected harness [6], then trains on fresh on-policy rollouts with task-specific rubric rewards and GRPO [7, 14]. The harness and rubrics remain fixed during training; historical interactions provide task definitions and diagnostic evidence rather than on-policy training samples.

The coupling is bidirectional: harness revisions shape training trajectories and task difficulty, while model updates change the effectiveness of inherited procedures. Background improvement proceeds alongside the online service. After re-evaluation, the updated model–harness pair returns to researchers, whose interactions initiate the next cycle. All harness and environment versions are retained for subsequent evolution. Thus, each outer cycle learns through an inner adaptation process and changes the model that participates in the next.

Our case studies examine real researcher interactions, harness revision with a fixed task model, and model learning with a fixed harness. The benchmark cases cover four task families from LAB-Bench and Biomni-Eval1: literature reading, database judgments, protocol troubleshooting, and gene and variant assessment [9, 10]. Holding one component fixed provides a focused view of changes in the other: the harness case measures first-response accuracy on feedback-accessible evaluation tasks, while the model case measures problem coverage on a common panel under H0. These studies connect the proposed framework to observable improvements in scientific task execution.

We release ScienceBuddy as an interactive research product, bringing scientific assistance and continual capability improvement into a shared workspace for researchers. This release makes our proposed paradigm available to the scientific community and takes a step toward **discovery intelligence**, where scientific agents evolve through sustained collaboration with the researchers they support.

Contributions. Our contributions are fourfold:

- **A released scientific research workspace.** We develop and release ScienceBuddy, an interactive product that helps researchers carry out scientific tasks by connecting researcher dialogue, executable analysis, inspectable artifacts, and a pluggable harness within a persistent workspace.
- **Interaction-grounded tasks and supervision.** We formulate a workflow for deriving executable tasks and evaluation rubrics from collaboration, with validated environment augmentation calibrated to current capabilities.
- **Recursive-in-recursive self-improvement.** We introduce a paradigm for model–harness co-design that couples evaluated harness evolution with rubric-supervised model reinforcement learning, returning the updated system to researchers for renewed interaction and adaptation.
- **Case-study evidence for procedural and model learning.** We examine real researcher interactions, fixed-model harness evolution, and model learning under a fixed harness, relating the proposed framework to improved scientific task execution and broader problem coverage.

2. ScienceBuddy

ScienceBuddy is an interactive scientific research workspace that brings evidence access, computational analysis, and methodological guidance into a single conversational workflow. Researchers can introduce questions together with their data, inspect the resulting analyses, and refine the work through subsequent exchanges. Built on this foundation, ScienceBuddy supports *recursive-in-recursive self-improvement*: an inner process revises and evaluates the agent's harness while keeping the task model fixed (Section 2.3), and an outer process applies continual reinforcement learning to trajectories generated under the evolving harness (Section 2.4). The updated model then returns to further harness adaptation, coupling improvements in working procedures with improvements in the model that executes them (Section 2.5). Figure 2 summarizes the coupled harness and model improvement process.

2.1. Scientific Workspace & Agent Harness

We first describe three system components: scientific tools and execution environments, agent execution and researcher interaction, and modular infrastructure with a pluggable harness. Figure 2 summarizes the scientific workspace and researcher interaction.

Scientific tools and execution environments. ScienceBuddy provides access to a catalog of 224 tools across 22 functional modules, spanning genomics, molecular and cancer biology, pharmacology, bioimaging, literature retrieval, and database queries. The runtime supports Python, R, and Bash execution, combining scientific libraries with data processing, statistical analysis, and visualization. Online database interfaces and a local data lake provide complementary access to biomedical evidence. Researcher-provided documents, tables, sequences, and images enter a persistent workspace that retains inputs, intermediate files, and generated outputs. Interface and environment details appear in Section S1.1. The scientific tool catalog and execution utilities are derived from Huang et al. [9].

Agent execution and researcher interaction. We follow a ReAct-style reasoning–action–observation loop [23], alternating reasoning, code or tool execution, and observation. Researchers submit questions, upload supporting data, and provide follow-up instructions through the Chat view. The Trajectory view presents the chronological execution record, an event timeline, and details of selected events. Compute and Results panels provide access to execution activity and generated artifacts. Conversation history and workspace files preserve task context across exchanges, allowing researchers to inspect the agent's work and request revisions. Section S1.5 illustrates both interface views.

Modular infrastructure and pluggable harness. ScienceBuddy separates the agent harness from the infrastructure that manages researcher interactions, task execution, and persistent workspaces. A common execution interface specifies the task context supplied to the harness and the responses and execution records returned to the platform. Alternative agentic harnesses can be integrated by implementing this interface, while sharing the same task-management and storage services. Within this architecture, instructions, skills, and selected context-management procedures constitute the editable components

of the harness. Recursive improvement revises these components while keeping the surrounding infrastructure fixed, allowing changes in scientific problem-solving procedures to be evaluated under consistent execution conditions (Section 2.3).

2.2. Interaction Formulation and Learning Signals

Interaction formulation. Let x denote a research request and its inputs, π_θ the task model, H the harness, and $h_t = (x, a_0, o_1, \ldots, a_{t-1}, o_t)$ the history, with $h_0 = (x)$. An action a_t is executable code, a tool call, or a researcher-facing response. The observation $o_{t+1} = (e_{t+1}, u_{t+1})$ records environment output or execution status e_{t+1} and an optional researcher reply u_{t+1}, with $u_{t+1} = \perp$ when absent. The harness constructs model context $C_H(h_t)$ from history, memory, skills, and tool descriptions. Allowing for a scheduled deterministic action $d_H(h_t)$, such as input inspection, the joint policy and trajectory are

$$\mu_{\theta,H}(a \mid h_t) = \begin{cases} \delta_{d_H(h_t)}(a), & \text{if a harness action is scheduled,} \\ \pi_\theta(a \mid C_H(h_t)), & \text{otherwise,} \end{cases} \tag{1}$$
$$a_t \sim \mu_{\theta,H}(\cdot \mid h_t), \qquad \tau = (x, a_0, o_1, \ldots, a_{T-1}, o_T).$$

Here δ denotes a point mass and T counts execution steps. A researcher-facing response may follow several tool steps; a tool observation alone does not constitute a researcher turn or user feedback.

From collaboration to tasks and rubrics. The collaboration record supplies two complementary artifacts: a self-contained task and its evaluation rubric (Figure 3). Related turns are consolidated around a scientific objective, with independently solvable objectives separated. The task instruction preserves the final requirements and inputs without importing the historical answer. Unlike earlier task-only packaging followed by expert annotation, the current workflow also derives the rubric from the full collaboration trajectory:

$$C(x) = \text{ConstructRubric}\big(\tau_x^{\text{collab}}; I_x, A_x\big), \tag{2}$$

where τ_x^{collab} is the source collaboration, I_x the reconstructed instruction, and A_x the required assets. Criteria cover task scope, methodological requirements, evidence, and expected artifacts. Conflicting requirements are resolved before scoring; historical answers and researcher approval are not automatically treated as scientific ground truth.

Harbor tasks for post-training. The task package combines the instruction, input assets, execution environment $\mathcal{E}_x$, and rubric:

$$\mathcal{P}_x = \big(I_x, A_x, \mathcal{E}_x, C(x)\big). \tag{3}$$

Instructions, configuration, assets, and rubric-based tests are organized as Harbor tasks [1]. The same tasks support two post-training routes: SFT retains rubric-qualified generated trajectories through rejection sampling, while RL collects fresh on-policy rollouts and uses rubric scores as rewards. Input and runtime checks establish executability; rubric-based checks and a fixed judge assess scientific requirements. The rubric remains fixed within each post-training stage.

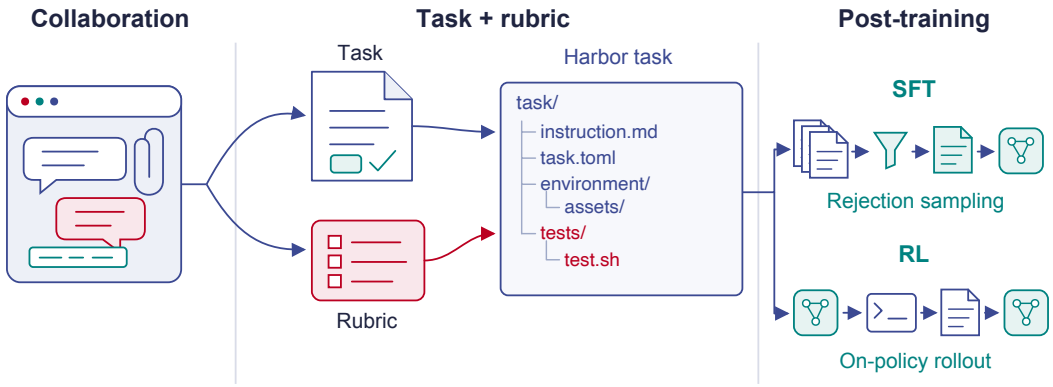

Figure 3 | Collaboration-derived Harbor tasks for post-training. The collaboration supplies both the task definition and rubric. In the schematic file tree, `instruction.md` defines the task, `task.toml` configures execution, `environment/` holds task assets, and `tests/test.sh` invokes rubric-based assessment. The resulting tasks support SFT through rejection sampling and RL through on-policy rollouts.

2.3. Inner Recursion: Feedback-guided Harness Improvement

At inner step j of outer cycle k, the active harness $H_{k,j}$ is the *parent*, and its proposed revision $\widetilde{H}_{k,j+1}$ is a *candidate*. An accepted candidate becomes the *child* $H_{k,j+1}$. Otherwise, the parent remains active.

Feedback-guided diagnosis. Within outer cycle k, the task-model parameters θ_k remain fixed. We use GPT-6 Astra as a separate, fixed auxiliary model for trajectory diagnosis and harness editing. It reviews recent trajectories and rubric evaluations, identifies unmet criteria, and cites the relevant actions and observations. Following evidence-based trajectory diagnosis [3], it maps these findings to a candidate procedural edit. Task-specific answers and newly supplied facts remain local to the task.

Harness revision. Let $E_{k,j}$ contain the selected trajectories, rubric feedback, and edit history for harness $H_{k,j}$. The auxiliary model proposes a bounded update,

$$\widetilde{H}_{k,j+1} = U(H_{k,j}, E_{k,j}; \theta_k). \tag{4}$$

Each proposal adds, removes, or revises one scoped skill, edits an instruction, or changes one exposed context setting, leaving other components unchanged [12, 22]. A schema check enforces the permitted edit scope and size budget. Tools, execution infrastructure, rubrics, and evaluators remain fixed. This is a procedural update; neither the task model nor the auxiliary model receives gradient updates.

Evaluation and recursive refinement. Parent and candidate are evaluated on identical development tasks, seeds, and execution budgets using frozen task rubrics. Let $\bar{S}_k(H)$ be the mean normalized rubric score and $\mathrm{Valid}(H)$ indicate compliance with the edit constraints. Write $\Delta_{k,j} = \bar{S}_k(\widetilde{H}_{k,j+1}) - \bar{S}_k(H_{k,j})$. The proposed acceptance rule is

$$H_{k,j+1} = \begin{cases} \widetilde{H}_{k,j+1}, & \mathrm{Valid}(\widetilde{H}_{k,j+1}) \ \wedge \ \Delta_{k,j} > 0, \\ H_{k,j}, & \text{otherwise.} \end{cases} \tag{5}$$

Evaluation includes previously successful tasks to account for regressions [13], and ties retain the parent. Rejected edits and score changes remain in the optimizer's history. The

selected harness then executes new training tasks, whose trajectories supply evidence for the next revision. Iteration continues until the proposal budget or outer collection boundary is reached. Development tasks are separate from policy-training tasks and the final held-out test set, which never informs editing or selection.

2.4. Outer Recursion: Continual Model Reinforcement Learning

Environment augmentation under an evolving harness. As the harness evolves, previously challenging tasks may become routine, reducing their value for further model training. We therefore calibrate environment difficulty through pilot execution with the current task model and selected harness. Following environment evolution [6], we augment researcher-derived tasks by varying scientific inputs and analysis conditions or extending dependencies between computational steps. The validated environments then supply fresh RL rollouts.

Task-adaptive rubric rewards. For each task x, a fixed rubric composer derives task-specific criteria from the source collaboration trajectory and its reconstructed objective, inputs, and required outputs (Section 2.2), following task-adaptive rubric construction [4]. The resulting rubric $C(x)$ combines task-specific correctness checks with relevant evidence and artifact requirements. Each criterion has a nonnegative importance weight $w_c(x)$, assigned before rollout evaluation, and a satisfaction score $v_c(x,\tau) \in [0,1]$. We use executable checks where available and a fixed judge for criteria requiring scientific interpretation [24]. Following rubric-based reward aggregation [7], the trajectory reward is

$$R_x(\tau) = \frac{\sum_{c\in C(x)} w_c(x)\, v_c(x,\tau)}{\sum_{c\in C(x)} w_c(x)}, \qquad \sum_{c\in C(x)} w_c(x) > 0. \tag{6}$$

Rubrics vary across tasks but remain fixed during optimization and paired harness evaluation. The terminal reward supplies a trajectory-level advantage shared across generated tokens. We use GRPO [14]; its objective and implementation details are given in Section S1.3.

Model updates and renewed harness adaptation. At outer cycle k, we maximize the expected trajectory reward under the selected harness:

$$\max_{\theta} J_k(\theta), \qquad J_k(\theta) = \mathbb{E}_{x\sim q_k}\mathbb{E}_{\tau\sim\pi_{\theta,H_k^\star}(\cdot|x)}\big[R_x(\tau)\big]. \tag{7}$$

Here q_k is the training-task distribution over the validated seed environments and augmented variants at outer cycle k, and $\pi_{\theta,H_k^\star}$ is the trajectory distribution induced by the task model under the fixed harness $H_k^\star$. The GRPO update yields θ_{k+1}. Because harness effectiveness depends on its interaction with the task model [11], we re-evaluate the selected harness under the updated model before deploying (θ_{k+1}, H_{k+1}), with $H_{k+1} = H_k^\star$. Researcher interactions with this pair provide evidence for the next inner-adaptation phase and outer update cycle (Section 2.5).

2.5. Coordinating Recursive-in-Recursive Improvement

Nested update schedule. ScienceBuddy serves researchers with model θ_k and harness H_k over a fixed collection interval. The resulting interactions and feedback initiate

a background update cycle, asynchronous with the online service: harness improvement proceeds with θ_k fixed, followed by model RL under the selected harness. After re-evaluation, the updated model–harness pair is deployed to support increasingly demanding research tasks. Subsequent researcher interactions provide the evidence for the next cycle (Algorithm 1).

Cross-cycle experience and re-evaluation. All harness versions and task environments are retained for subsequent evolution. Inherited harnesses are re-evaluated under the updated task model before deployment or reuse.

Algorithm 1 Recursive-in-recursive improvement with asynchronous online service

Require: Initial (θ_0, H_0), collection interval Δ, training environments $\mathcal{T}$, development tasks, inner budgets J_k, RL budgets, and outer count K

Initialize evidence buffer $\mathcal{B}$ and edit history $\mathcal{L}$; deploy (θ_0, H_0)
for $k = 0, \ldots, K-1$ **do**
  $E_k \leftarrow \mathrm{Collect}_\Delta(\theta_k, H_k)$; $\mathcal{B} \leftarrow \mathcal{B} \cup E_k$
  Background updates; the online service continues with (θ_k, H_k).
  $H_{k,0} \leftarrow H_k$; evaluate $\bar{S}_k(H_{k,0})$; $j \leftarrow 0$
  while $j < J_k$ and the inner execution budget remains **do**
    Append fresh task evidence under $(\theta_k, H_{k,j})$ to $\mathcal{B}$
    $E_{k,j} \leftarrow \mathrm{Read}(\mathcal{B}, \mathcal{L}; H_{k,j})$
    $\widetilde{H}_{k,j+1} \leftarrow U(H_{k,j}, E_{k,j}; \theta_k)$ ▸ Section 2.3
    Validate and, if valid, evaluate $\widetilde{H}_{k,j+1}$ under paired development conditions
    Select $H_{k,j+1}$ by Equation (5); record the decision in $\mathcal{L}$
    $j \leftarrow j + 1$
  end while
  $H_k^\star \leftarrow H_{k,j}$; $\mathcal{T}_k \leftarrow \mathrm{Augment}(\mathcal{T}; \theta_k, H_k^\star)$
  $\theta_{k+1} \leftarrow \mathrm{RLUpdate}(\theta_k; H_k^\star, \mathcal{T}_k, R)$ ▸ Section 2.4
  Use fresh batches $\mathcal{D}_{k,t}$, Equations (S3) and (6); retire batches after optimization.
  $H_{k+1} \leftarrow H_k^\star$; re-evaluate (θ_{k+1}, H_{k+1})
  Retain all harness and environment versions; $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_k$
  Deploy (θ_{k+1}, H_{k+1})
end for
return (θ_K, H_K)

3. Scientific Workspace and User Experience

Scientific scope. ScienceBuddy combines multimodal input, long-context agentic reasoning, and researcher interaction within a shared scientific workspace. Its document handling and execution interfaces support multiple scientific domains, while the current tools and data specialize in biomedicine. The following recorded session illustrates how researchers connect visual scientific material to target analysis, evidence retrieval, and further questions.

Multimodal input and evidence inspection. Researchers can supply documents, tables, biological sequences, and images alongside natural-language requests. In Figure 4, an uploaded immune-signaling diagram guides the identification of molecular targets and the organization of related drug and pathway knowledge. The response connects visual entities to an evidence table, distinguishing a retrieved PDE4/rolipram fragment from CD40 and AHR searches that returned no matches. The conversation, input composer,

and Compute panel bring the scientific material, response, and execution history into one inspectable view. Original interface captures appear in Section S1.5.

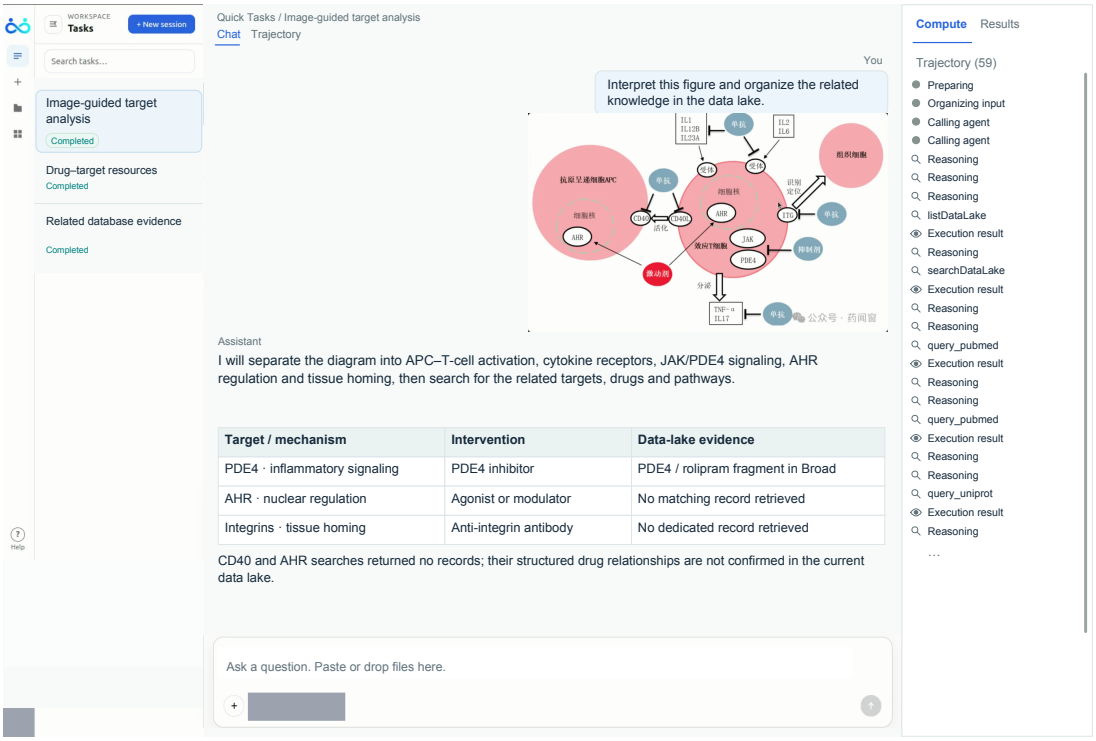

Figure 4 | A workspace for multimodal scientific analysis. A researcher supplies a scientific diagram and requests related knowledge. The Chat view connects visual interpretation to a structured target–evidence table, while the Compute panel exposes execution records. Retrieved evidence and gaps in the available data remain visible for researcher inspection. UI text and dialogue are reconstructed in English from the recording; uploaded figures retain their original appearance and language. Account and model identifiers are masked.

Long-context agentic reasoning. Figure 5 follows three image-based requests in a continuing session: an HMGCR Mendelian-randomization diagram, an Alzheimer's-related microglial network, and an immune-signaling diagram. The agent interprets each image through reasoning, retrieval, and synthesis; the later execution explicitly resumes the same session with prior exchanges available. The trace records repeated data-lake searches and literature/protein queries; the middle response instead uses model knowledge without a new database query.

Researcher interaction. The researcher directs the work by introducing new diagrams, changing the scientific focus, and explicitly requesting database evidence. Successive responses organize targets, distinguish pathways from cell-state markers, and identify data needed for further analysis. Retained dialogue and evidence support subsequent requests and the derivation of task objectives and evaluation criteria (Section 2.2).

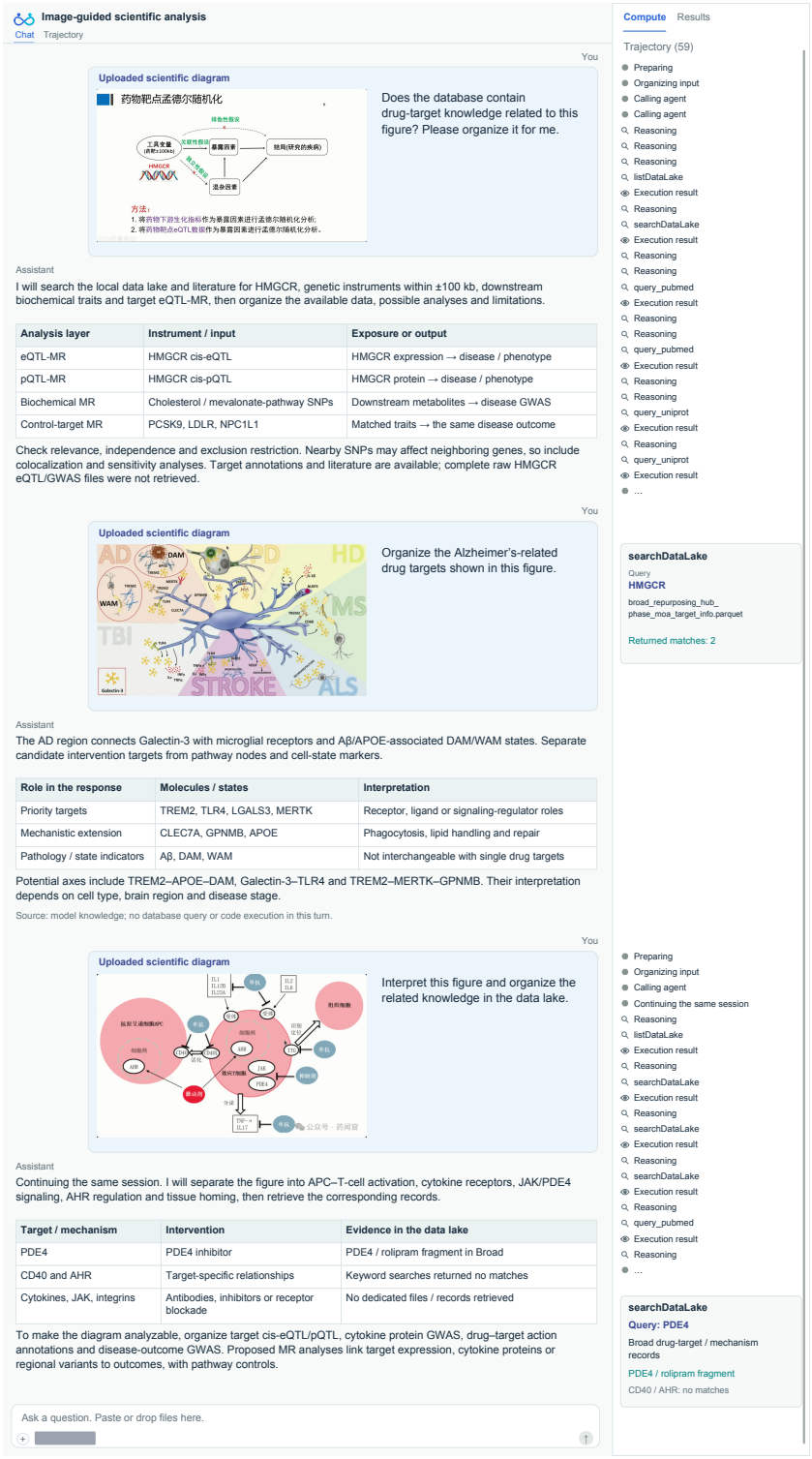

Figure 5 | **Multimodal input, long-context agentic reasoning, and researcher interaction.** Three successive image-based requests direct target analysis across a continuing scientific session. Uploaded diagrams, assistant interpretations, and evidence tables are paired with the recorded execution history, showing how researcher direction and retained context connect successive rounds of work. Proposed analyses are not executed experiments. English dialogue and UI are reconstructed from recorded moments; uploaded figures retain their original language, omitted events are marked, and identifiers are masked.

Researcher inspection through interface controls. Researcher interaction also includes navigation and inspection actions beyond conversational input (Figure 6). In the demonstration, the researcher opens an uploaded diagram at a larger scale, switches from Chat to Trajectory, and selects a tool event to inspect its metadata, input, and output. The selected UniProt event exposes an earlier HMGCR lookup while later requests remain in the same session. These controls let the researcher examine source material, follow the execution history, and revisit the basis of a response without starting a new conversation.

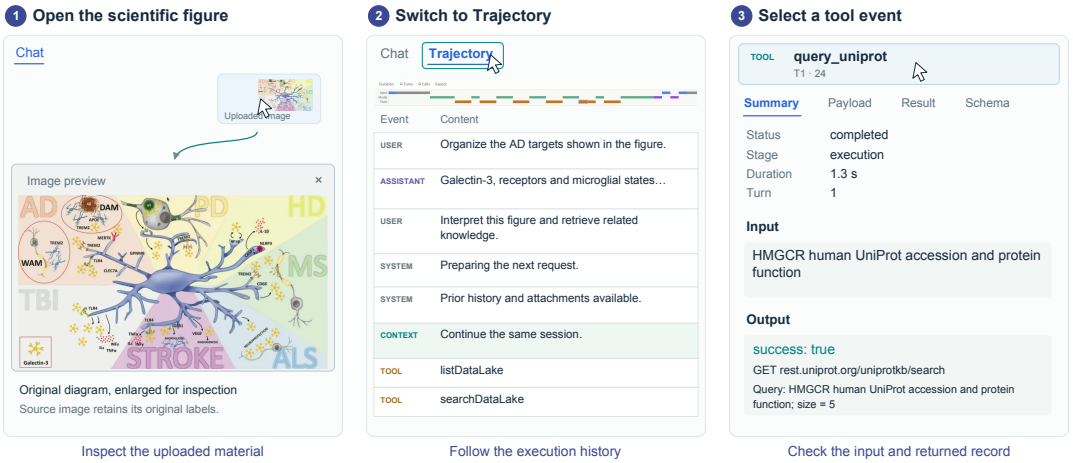

Figure 6 | Researcher inspection beyond the conversation. Opening an uploaded image reveals its scientific details; switching to Trajectory exposes the execution history; selecting a tool event opens its input, output, and metadata. The example revisits an earlier HMGCR protein lookup within the continuing session. English interface reconstructions highlight controls used in the recording; the uploaded diagram and timeline retain source pixels. Cursor markers indicate the inspected controls.

4. Case Studies

We present four distinct case studies of ScienceBuddy's scientific assistance and self-improvement. Each addresses a separate research question:

RQ1: Researcher interaction. How does researcher feedback guide scientific assistance and reveal task objectives and evaluation criteria? (Section 4.1)

RQ2: Coupled Recursive-in-Recursive improvement. Can alternating harness refinement and model learning sustain improvement across cycles and broaden scientific task performance? (Section 4.2)

RQ3: Harness adaptation. Can harness adaptation improve scientific task performance without changing model weights? (Section 4.3)

RQ4: Model learning. Can reinforcement learning expand scientific problem-solving capability under a fixed harness? (Section 4.4)

4.1. ScienceBuddy Interaction

Setup. We examine two real researcher interactions with deployed ScienceBuddy. Requests, supplied materials, agent responses, and subsequent researcher input support a qualitative assessment of scientific assistance and opportunities for reinforcement

learning (RL) task construction. Readers interested in the concrete researcher wording can consult Figure S1 in the appendix.

Refining a JAK1 investigation. A researcher asked ScienceBuddy to design a study of JAK1, immunotherapy outcomes, and the immune microenvironment in small-cell lung cancer using public single-cell transcriptomes and IMpower133 bulk RNA data. In response to the scope refinement (Figure S1a), ScienceBuddy organized a gene-specific plan with treatment-by-JAK1 interaction tests, patient-level expression summaries within cell types, and immune-state signatures. The plan assigned Seurat/Scanpy to single-cell analysis, UCell/AUCell to signature scoring, and CellChat/NicheNet to subsequent cell-communication analyses. This plan distinguished treatment-effect modification from prognosis and prioritized mechanistic follow-up.

Connecting evidence in an ARL4C study. A researcher requested a presentation connecting the background and results of an ARL4C study, then specified panel selection, conclusions, mechanism schematics, and speaker notes (Figure S1b). Using text and figure captions organized through Python/PyPDF2, ScienceBuddy linked candidate screening to cellular and molecular evidence. It highlighted depletion and conditional knockout comparisons for cellular attribution, blockade for functional dependence, and kinetic and rescue assays for molecular interpretation. Panel-selection rationales and notes linked each scientific claim to its supporting comparison.

From requests to task specifications. These cases illustrate how researcher requirements translate into task objectives, evaluation criteria, and required artifacts (Figure 7). The JAK1 refinement yields a study-planning objective whose criteria preserve gene-specific scope and place association analyses before mechanistic follow-up. The ARL4C request yields a presentation objective whose criteria link claims to supporting panels and comparisons, with conclusions and speaker notes accompanying the slide outline. Such task specifications provide the basis for the trajectory-derived rubrics and post-training tasks described in Section 2.2.

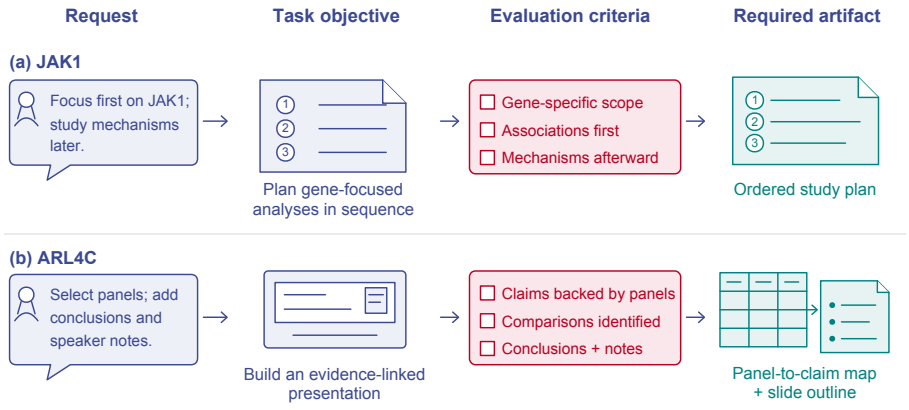

Figure 7 | From researcher requests to task specifications. (a) A JAK1 scope refinement defines an ordered, gene-focused study plan. (b) ARL4C presentation requirements define an evidence-linked presentation and panel-to-claim map. Requests are translated and abridged from real interactions; the task objectives, evaluation criteria, and required artifacts are illustrative derivations, not archived rubric packages or scored outputs.

4.2. Two-Cycle Recursive-in-Recursive Dynamics

Setup. Starting from Qwen3.5-4B and an initial scientific-agent harness, we run three successive co-evolution cycles, indexed by $k = 0, 1, 2$. In cycle k, harness refinement starts from (θ_k, H_k), keeps the model fixed, and performs 10 search steps to select $H_k^\star$ by validation accuracy. Model learning then performs 20 RL updates under the selected harness. The final checkpoint θ_{k+1} and selected harness $H_{k+1} = H_k^\star$ are carried into the next cycle, where inherited harnesses are reassessed under the updated model. This repeated exchange allows improvements in the model and harness to carry forward, supporting continued system improvement across successive cycles. Dataset and environment details are provided in Appendix S1.1; detailed experimental settings are deferred to the appendix.

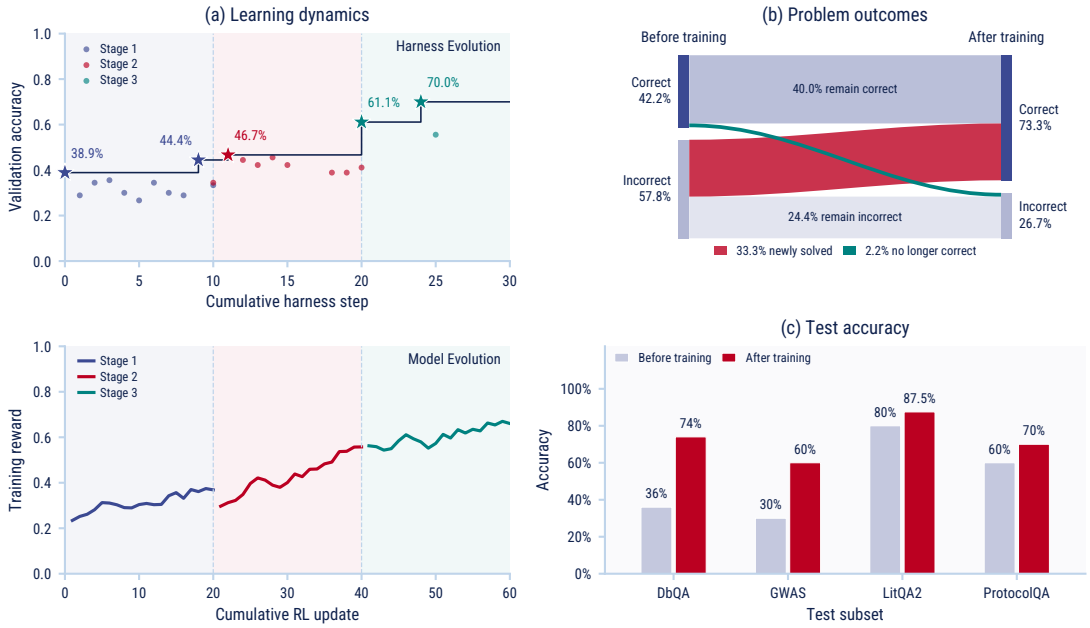

Figure 8 | Learning dynamics and evaluation across three RinR cycles. Each cycle comprises ten harness-evolution steps followed by twenty RL updates; colors identify cycles. (a) Circles show measured harness validation scores, including rejected candidates. Stars and annotations identify new historical bests. (b) Outcome transitions pair the initial and final systems on the same test sets. (c) Test accuracy by scientific task family. Harness selection uses a separate, fixed validation set.

Learning dynamics across cycles. Figure 8(a) shows consistent improvements within each of the three cycles. Harness refinement increases validation accuracy from 38.9% to 44.4%, 34.4% to 46.7%, and 61.1% to 70.0% in the first, second, and third cycles, respectively. Over the same cycles, mean training reward rises from 33.3% to 38.8%, 44.1% to 60.5%, and 57.8% to 69.8% between the first and second halves of each RL phase. These gains show that both harness refinement and model training continue to improve their respective metrics over repeated cycles.

Scientific task performance. Figures 8(b,c) summarize the improvement in held-out scientific task performance. Overall single-attempt test accuracy increases from 42.2% to 73.3%. Among all test problems, 33.3% transition from incorrect to correct, whereas 2.2% transition from correct to incorrect. The subset comparison shows gains across

all four task families. These results indicate that improvement extends to previously unsolved problems, broadening the system's scientific problem-solving capability.

The next two case studies evaluate harness adaptation and model learning independently, holding model weights or the harness fixed, respectively (Sections 4.3 and 4.4).

4.3. Harness Adaptation with a Fixed Model

Setup. We refine and select the harness on an *adaptation set*, then compare the selected and initial harnesses on a separate *validation set*. Tasks from LAB-Bench and Biomni-Eval1 [9, 10] cover literature reading, database judgments, protocol troubleshooting, and gene and variant assessment. Implementation details appear in Section S1.2.

Adaptation and validation performance. Figure 9a tracks first-response accuracy during harness adaptation: the fraction of tasks answered correctly on the first submission. Across 24 adaptation batches, the best observed batch accuracy reaches 75.0%. The selected harness is then evaluated on validation tasks, alongside the initial harness (Figure 9b). Validation accuracy increases from 31.1% to 51.1%, a gain of 20 percentage points with model weights fixed. This improvement demonstrates the effectiveness of revising the agent's working procedures beyond the tasks used for adaptation and selection.

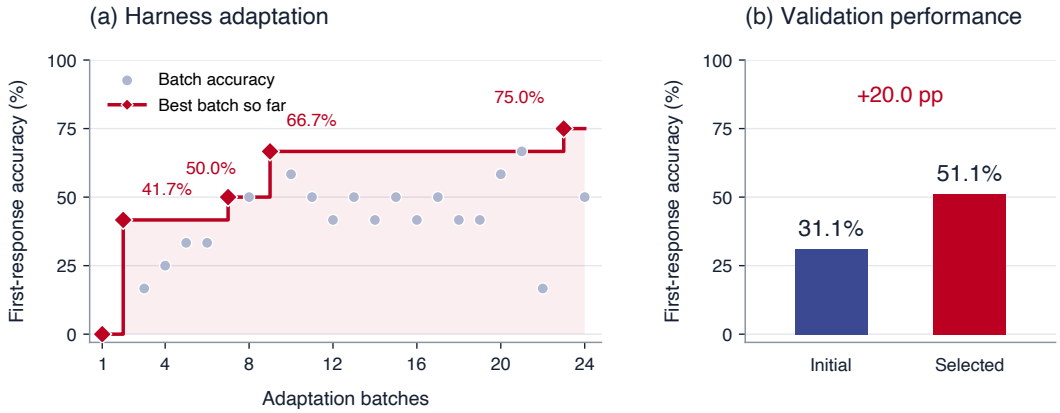

Figure 9 | Harness adaptation and validation performance at fixed model weights. (a) First-response accuracy across adaptation batches; the step curve tracks the best batch accuracy observed so far. Batches contain different tasks. (b) Validation accuracy of the initial harness and the harness selected on the adaptation set: 31.1% versus 51.1%, a gain of 20 percentage points. The validation set is used for this comparison, not harness selection.

Learned procedures. We inspect the selected harness to characterize the procedures retained from interaction. Its four instruction entries and nine scoped skills address Python execution, resource and schema inspection, bounded record lookup, and explicit answer submission. Task-specific procedures include gene-set membership checks, cytoband lookup, and database-specific evidence extraction. These procedures guide the agent in locating and checking scientific records, turning interaction evidence into reusable guidance for task execution. Section S1.2 describes the revisions and the limits of attributing gains to individual edits or feedback sources.

4.4. Model Learning with a Fixed Harness

Setup. We keep the initial harness fixed throughout training and compare the model before and after RL under the same evaluation budget. The learning algorithm and evaluation protocol appear in Section S1.3.

Learning dynamics and problem coverage. Training accuracy trends upward over approximately two hours of RL (Figure 10a). To assess whether learning also expands the range of solvable problems, we measure *problem coverage*: the fraction of test problems solved at least once within four attempts. Coverage increases from 48.3% before RL to 67.8% afterward (Figure 10b), a gain of 19.5 percentage points. With both the harness and attempt budget unchanged, the model solves a broader set of scientific problems, demonstrating the effectiveness of model learning as a distinct improvement mechanism.

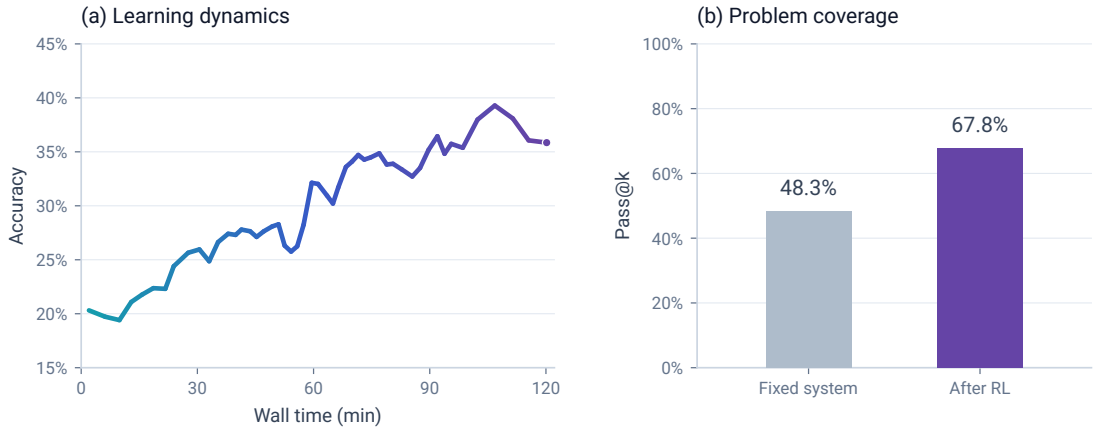

Figure 10 | Model learning and problem coverage under a fixed harness. (a) Training accuracy against elapsed time during model learning. (b) Problem coverage before and after RL, measured by pass@4 under the fixed harness and the same attempt budget. Coverage increases from 48.3% to 67.8%, indicating successful solutions to more distinct problems within the same attempt budget.

5. Related Work

Persistent experience in agents. Reflexion retains verbal lessons in episodic memory, GEPA searches over prompts using trajectory reflection, and ACE incrementally maintains contextual playbooks [2, 16, 27]. Meta-Harness extends search to harness code using prior candidates and execution records, while PILOT learns reusable procedures during live execution [11, 21]. These methods provide mechanisms for persistent procedural adaptation. ScienceBuddy studies how this adaptation generates experience for a second, model-level recursion.

Recursive self-improvement. The Darwin Godel Machine evolves an archive of agents, and Hyperagents makes the meta-level modification procedure part of the editable program [25, 26]. SEAL generates data and update directives for parameter adaptation [28]. ScienceBuddy instead studies a nested dependency between repeated harness adaptation and repeated task-model learning. Its reflector remains fixed, so improved task performance does not imply that the improvement mechanism itself has become stronger.

Learning from interaction. OpenClaw-RL extracts evaluative and directive signals from the states following agent actions, including user replies [19]. RLAnything jointly adapts environments, policies, and reward models [20]. ScienceBuddy studies how these learning processes interact with an evolving harness. User feedback guides procedural revision, while task verification supervises policy trajectories generated under the active harness. The learned model then returns to the next inner process, changing the conditions for further procedural adaptation.

6. Conclusion

ScienceBuddy provides an interactive scientific workspace in which researcher collaboration can inform both working procedures and model learning. Its recursive-in-recursive framework connects evaluated harness refinement with rubric-supervised model updates, returning the updated system to further scientific interaction. The case studies illustrate the complementary contributions of these components: researcher requests and follow-up requirements define scientific tasks and assessment criteria; harness revision improves first-response accuracy with the task model fixed; and model learning expands problem coverage under the initial harness. These findings support the framework's procedural and model-learning mechanisms and provide a basis for studying their coordination across continued researcher collaboration.

References

[1] Harbor: Task Structure. https://www.harborframework.com/docs/tasks. Accessed September 9, 2026.

[2] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv preprint arXiv:2507.19457*, 2025. doi: 10.48550/arXiv.2507.19457. URL https://arxiv.org/abs/2507.19457.

[3] Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. AgentRx: Diagnosing AI Agent Failures from Execution Trajectories. *arXiv preprint arXiv:2602.02475*, 2026. URL https://arxiv.org/abs/2602.02475.

[4] Liang Ding. AdaRubric: Task-Adaptive Rubrics for Reliable LLM Agent Evaluation and Reward Learning. *arXiv preprint arXiv:2603.21362*, 2026. URL https://arxiv.org/abs/2603.21362.

[5] Tianyu Fan and Chao Huang. HELIX: Model-Harness Co-evolution for Recursive Self-Improvement. *arXiv preprint arXiv:2608.13951*, 2026. URL https://arxiv.org/abs/2608.13951.

[6] Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiang Zhou, Jiangtao Guan, Jincheng Liu, Yun Yang, Dingxin Hu, Zhuo Han, Xing Wu, Feng Zhang, and Lilin Wang. Environment Evolution for Terminal Agents. *arXiv preprint arXiv:2609.04128*, 2026. URL https://arxiv.org/abs/2609.04128.

[7] Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. *arXiv preprint arXiv:2507.17746*, 2025. URL https://arxiv.org/abs/2507.17746.

[8] Prannay Hebbar, Yogendra Manawat, Samuel Verboomen, Alesia Ivanova, Selvam Palanimalai, Kunal Bhatia, and Vignesh Baskaran. SIA: Self Improving AI with Harness & Weight Updates. *arXiv preprint arXiv:2605.27276*, 2026. URL https://arxiv.org/abs/2605.27276.

[9] Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Yin Di, et al. Biomni: A General-Purpose Biomedical AI Agent. *bioRxiv*, 2025. doi:

10.1101/2025.05.30.656746. URL https://www.biorxiv.org/content/10.1101/2025.05.30.656746v1.

[10] Jon M. Laurent, Joseph D. Janizek, Michael Ruzo, Michaela M. Hinks, Michael J. Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D. White, and Samuel G. Rodriques. LAB-Bench: Measuring Capabilities of Language Models for Biology Research. *arXiv preprint arXiv:2407.10362*, 2024. doi: 10.48550/arXiv.2407.10362. URL https://arxiv.org/abs/2407.10362.

[11] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses. *arXiv preprint arXiv:2603.28052*, 2026. doi: 10.48550/arXiv.2603.28052. URL https://arxiv.org/abs/2603.28052.

[12] Haoyue Liu, Zhichao Wang, Yongxin Guo, Haoran Shou, and Xiaoying Tang. Adaptive Prompt Structure Factorization: A Framework for Self-Discovering and Optimizing Compositional Prompt Programs. *arXiv preprint arXiv:2604.06699*, 2026. URL https://arxiv.org/abs/2604.06699.

[13] Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. SkillGen: Verified Inference-Time Agent Skill Synthesis. *arXiv preprint arXiv:2605.10999*, 2026. URL https://arxiv.org/abs/2605.10999.

[14] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

[15] Woong Shin, Craig A. Bridges, Marshall T. McDonnell, and Rafael Ferreira da Silva. Fantastic Scientific Agents and How to Build Them: AgentBuild for Rietveld Refinement. *arXiv preprint arXiv:2606.12834*, 2026. URL https://arxiv.org/abs/2606.12834.

[16] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023. doi: 10.48550/arXiv.2303.11366. URL https://arxiv.org/abs/2303.11366.

[17] David Silver and Richard S. Sutton. Welcome to the Era of Experience. Preprint of a chapter for *Designing an Intelligence*, MIT Press, 2025. URL https://storage.googleapis.com/deepmind-media/Era-of-Experience%20/The%20Era%20of%20Experience%20Paper.pdf.

[18] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable Code Actions Elicit Better LLM Agents. *arXiv preprint arXiv:2402.01030*, 2024. doi: 10.48550/arXiv.2402.01030. URL https://arxiv.org/abs/2402.01030.

[19] Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. OpenClaw-RL: Train Any Agent Simply by Talking. *arXiv preprint arXiv:2603.10165*, 2026. doi: 10.48550/arXiv.2603.10165. URL https://arxiv.org/abs/2603.10165.

[20] Yinjie Wang, Tianbao Xie, Ke Shen, Mengdi Wang, and Ling Yang. RLAnything: Forge Environment, Policy, and Reward Model in Completely Dynamic RL System. *arXiv preprint arXiv:2602.02488*, 2026. doi: 10.48550/arXiv.2602.02488. URL https://arxiv.org/abs/2602.02488.

[21] Yang Xiao, Yusong Sun, Haoyi Wu, Wenyang Hui, Wen Da, Zhaokai Luo, Mu Chuan, Yao Hu, Wenjie Li, and Chengyue Jiang. PILOT in the Loop: Live Self-Improvement for Long-Horizon Agents. *arXiv preprint arXiv:2608.26530*, 2026. doi: 10.48550/arXiv.2608.26530. URL https://arxiv.org/abs/2608.26530.

[22] Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue-Ting Yang, and Chong Luo. SkillOpt: Executive Strategy for Self-Evolving Agent Skills. *arXiv preprint arXiv:2605.23904*, 2026. URL https://arxiv.org/abs/2605.23904.

[23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

[24] Ya-Qi Yu, Hao Wang, Fangyu Hong, Xiangyang Qu, Gaojie Wu, Qiaoyu Luo, Nuo Xu, Huixin Wang, Wuheng Xu, Yongxin Liao, Zihao Chen, Haonan Li, Ziming Li, Dezhi Peng, Minghui Liao, Jihao Wu, Haoyu Ren, and Dandan Tu. Reinforcement Learning with Robust Rubric Rewards. *arXiv preprint arXiv:2605.30244*, 2026. URL https://arxiv.org/abs/2605.30244.

[25] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954*, 2025. doi: 10.48550/arXiv.2505.22954. URL https://arxiv.org/abs/2505.22954.

[26] Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026. doi: 10.48550/arXiv.2603.19461. URL https://arxiv.org/abs/2603.19461.

[27] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. *arXiv preprint arXiv:2510.04618*, 2025. doi: 10.48550/arXiv.2510.04618. URL https://arxiv.org/abs/2510.04618.

[28] Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-Adapting Language Models. *arXiv preprint arXiv:2506.10943*, 2025. doi: 10.48550/arXiv.2506.10943. URL https://arxiv.org/abs/2506.10943.

Appendix

S1. Implementation Details

This appendix connects the case studies to the scientific workspace and recursive-in-recursive framework described in the main text. We distinguish the fixed-model harness-evolution run from the model-learning case under a fixed harness. The role specifications describe the information boundaries and procedural responsibilities of these components; the policy objective shows how model learning fits within the full recursive procedure.

S1.1. Datasets and Environments

Task composition. The 895-task collection contains 96 LitQA2, 511 DbQA, 108 ProtocolQA, and 180 GWAS tasks. LitQA2 and ProtocolQA each have one subtopic, DbQA has ten, and GWAS has four, for 16 subtopics in total. Table S1 reports the number of tasks in each subtopic and the totals for each task family.

Table S1 | Task counts across four scientific task families and 16 subtopics, totaling 895 tasks.

Family	Subtopic	Count
LitQA2	Scientific literature reading	96
DbQA	Disease–gene associations	39
	Gene location	40
	miRNA targets	40
	Mouse tumor gene sets	80
	Oncogenic signatures	40
	Transcription-factor binding (GTRD)	40
	Variant annotation: single sequence	80
	Variant annotation: multiple sequences	72
	Vaccine-response gene sets	40
	Viral protein interactions	40
	Subtotal	511
ProtocolQA	Experimental protocol troubleshooting	108
GWAS	Causal genes: GWAS Catalog	42
	Causal genes: Open Targets	45
	Causal genes: PharmaProjects	50
	Variant prioritization	43
	Subtotal	180
Total	16 subtopics	895

Roles of the task sets. In the standalone harness case study, adaptation conversations guide procedural revisions and harness selection. The initial harness and the harness selected on this adaptation set are subsequently compared on a separate validation set. The model-learning case compares two model checkpoints on the same panel under the initial harness, with four attempts per problem. Dataset counts describe the task inventory; the 288 conversations reported for harness adaptation describe the executed adaptation stream. The real researcher interactions in Section 4.1 separately illustrate how scientific requests and follow-up requirements can define task contexts and rubrics.

S1.2. Harness Evolution

Models and schedule. The reported harness run uses a fixed Qwen3.5-4B task model. A fixed Qwen3.8-27B helper supports bounded user simulation and feedback interpretation, while GPT-6 Astra proposes harness edits. After each batch of 12 task conversations, user feedback and execution evidence guide an update, giving 24 updates over 288 adaptation conversations. These conversations provide the feedback used for harness refinement; validation tasks are reserved for comparing the initial and adaptation-selected harnesses.

Editable procedures. The general harness interface permits instruction, skill, and selected context-management updates (Section 2.1). The reported case study restricts adaptation to instruction and scoped-skill text in a single-file harness. The execution loop, Python tool interface, context handling, input-inspection settings, submission checks, and budgets remain fixed. H0 starts without added instruction or skill entries; H24 contains four instruction entries and nine scoped skills. Each rollout executes a fixed source snapshot, with its harness version retained in the trajectory. The role specification below preserves the same instruction/skill-only edit boundary.

Revision and checkpoint selection. Proposed instruction or skill revisions pass component validation and a fixed execution preflight. The best-performing harness on the adaptation set is selected for the validation comparison; validation scores do not guide revision or checkpoint selection. On the validation set, the selected harness achieves 51.1% correct, compared with 31.1% for the initial harness. This standalone experiment's selection protocol is distinct from validation-based harness selection in the coupled-cycle experiment.

Simulated feedback. The simulator receives correctness and submission-status verdicts and selects a permitted reply for the task family. Correct answers receive confirmation, missing submissions receive a format request, and incorrect answers receive a procedural check or revision request. The reply set excludes the correct option, identifier, and numerical answer. The feedback interpreter sees the reply and its public conversation context, but not the private verdict or answer. It classifies the reply as acceptance, correction, new information, new requirement, or ambiguity and records a supporting quote and a diagnostic score $q_t \in \{-1, 0, +1\}$. These bounded replies provide controlled procedural feedback; the real researcher interactions in Section 4.1 illustrate the broader collaboration setting.

Diagnostic feedback and optimization reward. The score q_t records how a follow-up relates to the preceding response and supports harness diagnosis. The field named

Table S2 | Information available to each role. Public context includes the task, supplied assets, and legitimately observed tool outputs. Diagnostic feedback supports harness revision; the separately evaluated trajectory reward supports policy optimization. The simulator and feedback interpreter use the fixed helper model, while GPT-6 Astra performs harness diagnosis and editing.

Role	Public context	User next reply	Private answer	Verifier verdict
Task policy	Visible prefix	After response	Hidden	Through bounded user feedback
User simulator	Review context	Produces reply	Hidden	Correctness and submission status
Feedback judge	Prior context	Observed reply	Hidden	Hidden
Harness reflector	Parent's public trace	Observed reply	Hidden	Evaluation summaries
Scientific verifier	Required output	Not required	Private access	Produces verdict
Policy learner	Recorded policy input	Not backfilled	Not in prompt	Trajectory reward $R_x(\tau)$

`reward` in the interpreter specification denotes this diagnostic score. Policy optimization instead uses the separately evaluated trajectory reward $R_x(\tau)$ in Equation (6). Researcher feedback can inform a task's objectives and rubric before evaluation; it does not replace assessment of the resulting rollout against those criteria. The GRPO advantage below is defined from $R_x(\tau)$, not by substituting the interpreter's ternary score. Table S2 summarizes these information boundaries.

Reflection record and interpretation. GPT-6 Astra receives recent task trajectories, active procedures, rubric feedback, and relevant edit history, excluding private answers and evaluator internals. Each diagnosis links an unmet criterion to supporting actions or observations and a proposed procedural edit. The optimizer records the parent, candidate, model and environment versions, evaluation conditions, and acceptance decision. Rejected edits remain available for later diagnosis; stored versions are a history of decisions, not a frontier for parent sampling. The learning curves describe the combined effect of successive procedural revisions at fixed model weights. Effects of individual skills and feedback sources are not separately isolated.

Role prompt specifications. The following concise specifications explain the task interface, information boundaries, and edit scope of the harness-evolution case study. They are expository descriptions of the roles rather than byte-for-byte archived request payloads. Concrete requests also supply task inputs, conversation records, permitted replies, the parent harness, and the runtime's edit schema. Private reference answers remain outside the policy and proposer inputs.

Task policy.

> You are a scientific assistant running in Science Buddy. You can reason and use Python in a persistent REPL. Public inputs are in /workspace/assets. The original task, including sequences, is in /workspace/assets/task_prompt.txt. Read long sequences from that file instead of copying them into generated code. Use the exact public filenames listed below. Python code must print results; bare expressions are not displayed. Scientific tool/data descriptions are in /opt/scitrace/TOOLS.md and /opt/scitrace/DATA.md. The available frozen data lake is mounted read-only at /opt/data/biomni_data/data_lake. Check which

files and records exist before claiming database evidence. For a tool call, output one <execute>Python code</execute> block and wait for its result. Otherwise, reply to the researcher with a brief explanation and one <answer>value</answer> tag. Do not claim to have inspected evidence or executed code unless you actually did so. Respond to the researcher's next reply, revising your work when warranted.

User simulator.

Role-play a researcher reviewing the assistant's ACTUAL response. This is a BOUNDED, REFERENCE-ASSISTED user simulator, not unrestricted expert feedback or a human trace. Choose the most useful and applicable reply from allowed_replies based on the conversation. The options request checks or confirm completion; none identifies the correct task answer. Do not add scientific claims, candidate names, numerical results or facts outside the allowed replies. The private correctness verdict concerns the selected answer, not every sentence of the explanation. Return JSON with reply equal to one allowed reply and done equal to answer_correct.

Feedback interpreter.

You interpret feedback for trajectory diagnosis in an interactive scientific assistant. Use the user's NEXT REPLY as evidence about the assistant's PRECEDING response. You do not receive a reference answer or terminal verifier score. Do not guess one. Score +1 for explicit acceptance/confirmation; -1 for a correction or request to redo caused by an error, omission or unmet prior requirement; 0 for new requirements, newly supplied facts, unrelated follow-ups or insufficient evidence. A successful tool call is not user approval. A request to recheck or revise the same answer, or to supply an answer format already requested, is a correction (-1), not positive progression or a new requirement. Judge what the feedback says, not whether the user is scientifically correct. Return ONLY JSON with reward (-1,0,1), feedback_type (acceptance,correction,new_information, new_requirement,ambiguous), evidence (an EXACT substring of the user's reply), and hint (a brief reusable improvement direction, empty when not supported). The reward field is the diagnostic feedback score, not the trajectory reward used for policy optimization.

Harness proposer: instruction/skill-only edits.

Improve the scientific assistant's instructions or scoped skills using the supplied parent harness and interaction evidence. Identify an unmet criterion, cite the relevant actions or observations, and propose one bounded procedural change: revise an instruction, or add, remove, or revise one scoped skill. Preserve all non-target entries and runtime settings. Return the revised harness using the supplied runtime schema and edit constraints; retain existing skills unless one is the target of the proposed change. Do not change context-history settings, input-inspection settings, tools, execution infrastructure, submission checks, budgets, rubrics, or evaluators. A complete serialized harness represents the local edit, not permission to rewrite every component. Avoid repeating rejected edits without new supporting evidence. Do not encode task-specific answers, numerical results, or sample IDs. A successful tool call does not prove scientific correctness; newly supplied information is not necessarily an error.

S1.3. Reinforcement Learning

Case-study configuration. The task backbone is Qwen3.5-4B. In Section 4.4, the initial harness H0 remains fixed throughout model training and evaluation. Both model checkpoints are evaluated on the same problems with four attempts per problem, so the before/after comparison examines model learning under a common procedural interface. GPT-6 Astra is the diagnosis/editor model for harness adaptation (Section S1.2); this fixed-H0 case does not invoke a new harness-adaptation phase. It illustrates the

model-learning component that can be coordinated with harness refinement in the full framework.

Evaluation measure. Training accuracy counts correctly solved attempts. Evaluation coverage, measured by pass@4, counts a problem once if at least one of its four attempts succeeds. The latter compares the breadth of solved problems under an equal attempt budget and is distinct from the first-response accuracy used in the harness case. The reported coverage rises from 48.3% to 67.8% under H0. The objective below formulates this model-learning step within the recursive framework.

Fresh rollout groups. We express the model-learning component using the notation of the general recursive framework. During outer stage k, the selected harness $H_k^\star$ remains fixed while the task model is optimized. The model-only case in Section 4.4 holds this harness at H0 throughout its comparison. The case study instantiates a fixed-harness model-learning step, while Section 2.4 describes how selected harnesses and validated task environments can be incorporated across cycles. For each rollout batch, a frozen copy π_{old} of the current task policy generates G trajectories per task. Records associate the model inputs, generated tokens, behavior log probabilities, task and rubric versions, and harness identifier with each trajectory. Fresh rollout groups supply the objective below; historical researcher interactions instead support task definition and procedural diagnosis. When a batch is reused for several optimizer passes, probability ratios remain relative to its original collection policy.

Group-relative policy objective. The GRPO formulation [14] uses token-level averaging. For trajectory i, let $r_i = R_{x_i}(\tau_i)$ denote its evaluated trajectory reward, distinct from the diagnostic feedback score q_t, and let $\mathcal{G}(i)$ contain trajectories generated for the same task under the same harness and rubric. The group-relative advantage is

$$\widehat{A}_i = \frac{r_i - \operatorname{mean}_{j\in\mathcal{G}(i)} r_j}{\operatorname{std}_{j\in\mathcal{G}(i)} r_j + \delta}, \qquad \delta > 0. \tag{S1}$$

All generated tokens in a trajectory share this advantage. Groups with identical rewards have zero policy-gradient advantage. Researcher messages, tool outputs, task instructions, and deterministic harness actions are excluded from the optimized tokens.

For generated token $b_{i,\ell}$ and its actual context $c_{i,\ell}$, define

$$\rho_{i,\ell}(\theta) = \frac{\pi_\theta(b_{i,\ell} \mid c_{i,\ell})}{\pi_{\text{old}}(b_{i,\ell} \mid c_{i,\ell})}. \tag{S2}$$

For a minibatch $\mathcal{M}$ of complete groups, with L_i generated tokens in trajectory i, the objective is

$$\mathcal{J}_k^{\text{GRPO}}(\theta) = \frac{1}{\sum_{i\in\mathcal{M}} L_i} \sum_{i\in\mathcal{M}} \sum_{\ell=1}^{L_i} \Big[\min\{\rho_{i,\ell}(\theta)\widehat{A}_i, \operatorname{clip}(\rho_{i,\ell}(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_i\} - \beta \widehat{d}_{i,\ell}(\theta) \Big]. \tag{S3}$$

Here $\epsilon > 0$ controls clipping and $\beta \geq 0$ weights the sampled KL surrogate. The reference policy π_{ref} is a frozen copy of the task model at the start of the outer RL stage. Writing

$z_{i,\ell} = \pi_{\text{ref}}(b_{i,\ell} \mid c_{i,\ell}) / \pi_\theta(b_{i,\ell} \mid c_{i,\ell})$, the surrogate is $\widehat{d}_{i,\ell} = z_{i,\ell} - \log z_{i,\ell} - 1$. This sampled quantity is evaluated on behavior-policy tokens; it is not asserted to be an exact KL divergence under an updated policy.

The trajectory reward is computed after the rollout from its outputs and task-relevant execution evidence, with rubric and evaluator parameters fixed during optimization. It does not backfill later evaluation information into the contexts that generated earlier tokens. In the model-learning case study, both checkpoints are evaluated under H0 using the common pass@4 protocol. In the full recursive framework, the resulting checkpoint θ_{k+1} returns to harness re-evaluation and deployment, and subsequent researcher interactions initiate the next cycle (Section 2.5). The same policy-learning formulation thus serves the fixed-harness comparison and provides the model-update step of the full recursive procedure.

S1.4. Concrete Researcher Inputs

The following excerpts reproduce the scope refinement and presentation requirements discussed in Section 4.1. They provide the concrete researcher wording underlying the illustrative task transformations in Figure 7.

(a) JAK1: study design

Scope refinement

Focus first on JAK1 itself: examine associations with immunotherapy outcomes, immune cell types, and immune signatures. Investigate upstream and downstream regulation and cell interactions afterward.

(b) ARL4C: evidence synthesis

Follow-up request

Organize the manuscript and figures, select key panels, provide highlights and a one-sentence conclusion for each results slide, develop consistent mechanism schematics, and write speaker notes.

Figure S1 | Researcher input for RL task construction. Scope refinements and follow-up requests provide task objectives and evaluation criteria. Both prompts are translated and abridged from real interactions.

S1.5. User Interface and Researcher Interaction

Chat and task management. The Chat view combines a task sidebar, a conversational workspace, and panels for execution activity and generated results (Figure S2). Researchers can create or revisit a task, choose a starter prompt, or enter a question directly. The input composer accepts pasted or uploaded files and supports follow-up instructions within the same conversation. The Compute and Results tabs provide access to analysis activity and resulting artifacts.

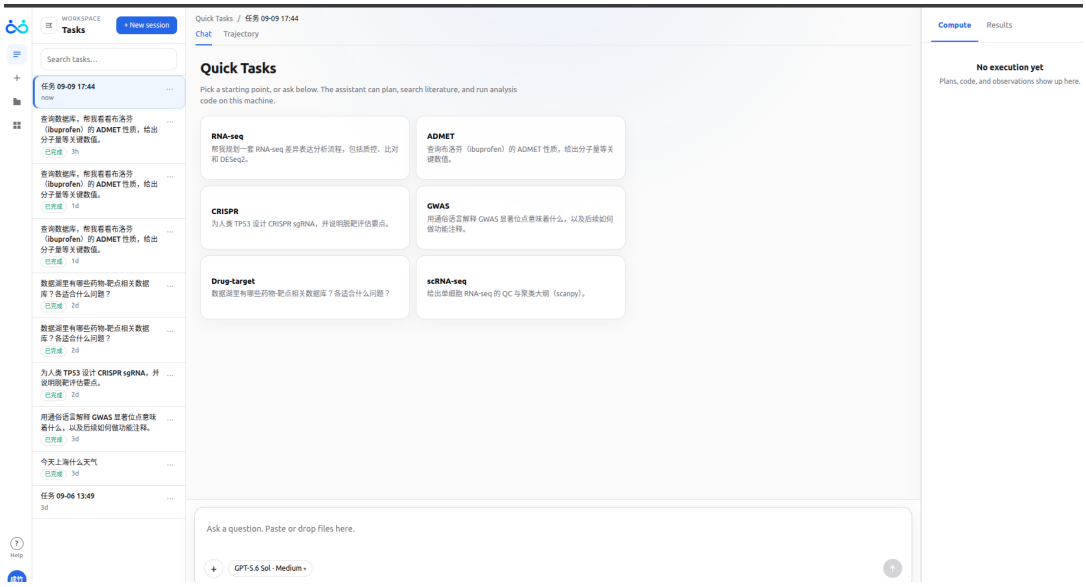

Figure S2 | Chat view in ScienceBuddy. The task sidebar appears on the left, starter prompts and the conversation area in the center, and Compute and Results tabs on the right. The input composer supports questions, file attachments, and model selection. This screenshot shows the initial task view before execution.

Trajectory inspection. The Trajectory view exposes the ordered record of a task, including user messages, system events, context summaries, tool calls, and assistant responses (Figure S3). A timeline separates input, model, and tool activity. Selecting an event opens a detail pane with Summary, Payload, and Result tabs for inspecting its recorded content. Search and export controls support reviewing the record, while the conversation composer remains available for subsequent input.

Figure S3 | Trajectory view in ScienceBuddy. The timeline and event record expose the progression of an analysis, and the right-hand pane displays details of a selected event. The screenshot shows a recorded compound-property query, its tool activity, subsequent dialogue, and a selected context entry.

Organizations

[1]PhAI Labs

[2]Department of Hepatobiliary Surgery and Transplantation, Liver Cancer Institute, Zhongshan Hospital, Fudan University

[3]State Key Laboratory of Genetics and Development of Complex Phenotypes

[4]Fudan University

[5]Shanghai Academy of Natural Sciences

[6]Shunwei Capital

[7]University of Oxford

[8]Stanford University

[9]Princeton University