RSI-WILD: BENCHMARKING AI DATA RESEARCHERS WITH IN-THE-WILD AGENTIC TRAJECTORIES FOR RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement (RSI) envisions AI systems that can help improve future generations of models. Recent benchmarks evaluate research agents on data-centric post-training tasks, but rarely expose them to the mixed-quality trajectory pools encountered in real-world data research. We introduce RSI-WILD, a benchmark for evaluating data research agents in this more realistic setting. RSI-Wild defines 23 error types across six categories and constructs execution-grounded failed trajectories, which are mixed with verified successes to form controlled mixed-quality pools. Our experiments reveal a **utility–reliability gap**. Frontier agents recover substantial value from noisy data, but the strongest reaches only 21.7% on the software engineering task, compared with 29% for human researchers. They fail to consistently produce better models across iterations, use research time inefficiently, and have difficulty translating external resources into concrete experiments. Error-level analysis further shows that agents more readily address errors with explicit contradictory evidence, while often overlooking locally coherent failures that require reasoning about missing actions or verification. Together, these findings suggest that current agents can make useful data improvements, but still struggle to turn feedback and external knowledge into sustained performance gains.

1 INTRODUCTION

Recursive self-improvement (RSI) asks whether models can take over the research loop of designing better training methods, running experiments, evaluating what works, and iterating toward the next generation of models (Chen et al., 2026a). Automating this loop could shift the scaling of research from human researcher capacity toward computational resources. Within this loop, training data plays a central role because what a model is trained on directly shapes what it can do. A growing line of work has therefore focused specifically on data-centric research (Kang et al., 2026; Liang et al., 2026; Meng et al., 2026). These benchmarks fix the training pipeline and evaluation protocol, while granting agents control only over the training data and evaluating their research decisions by the performance of the resulting model.

However, existing benchmarks share the following limitations: **(1) Realistic failed trajectories are not provided.** Existing benchmarks largely take one of two starting points: agents either reuse and curate existing public training data, as in PostTrainBench (Rank et al., 2026) and Curation-Bench (Kang et al., 2026), or construct new training data from seed tasks or raw sources, as in RSIBench-Data (Meng et al., 2026) and DataPrep-Bench (Liang et al., 2026). Neither starts from a pool of realistic failed trajectories that researchers must diagnose and improve. In practice, however, trajectories collected from humans or generated by capable models can contain both useful experience and realistic failures, making it necessary to decide what to retain, repair, or discard (Cao et al., 2026; DeepSeek-AI, 2025; Team, 2025a). **(2) Data operations are not failure-aware.** Existing benchmarks may expose trajectories, verifier outcomes, or other feedback, but do not provide controlled error-type annotations. Without such ground truth, they cannot directly attribute researcher behavior to specific failures and largely rely on aggregate downstream performance. This matters because aggregate downstream performance can hide qualitatively different research behavior: two researchers may achieve similar scores while detecting and correcting very different classes of failures.

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.

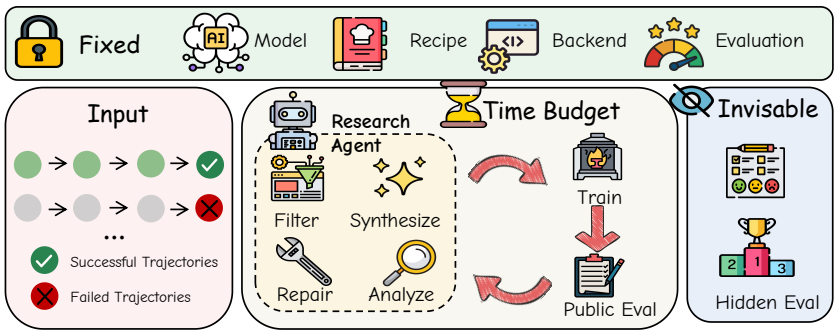

Figure 1: The automated data-research pipeline.

Therefore, we introduce RSI-WILD, a benchmark for evaluating AI data researchers on controlled mixed-quality trajectory pools. Drawing on prior analyses of failed trajectories in LLM agents (Liu et al., 2026; Zhu et al., 2026), we define a taxonomy of 23 error types across six categories. As illustrated in Figure 2, we inject these errors into successful trajectories and re-roll out the modified trajectories in their original environments using the same rollout models. This produces realistic failed trajectories with known error types and enables controlled analysis of which failures researchers recognize, overlook, and intervene on.

Our results reveal a **utility–reliability gap**. Frontier agents achieve meaningful improvements but still lag behind human researchers: the best agent reaches 21.7% on the software engineering task, compared with 29% for human researchers. We also find that agents fail to achieve consistent model improvements across iterations and are poor budget allocators, often stopping early. We further compare agents provided with research papers, skills, and GitHub repositories. Across these settings, agents devote substantial effort to browsing and verifying the provided resources but translate relatively few insights into concrete experiments. Error-level analysis reveals that agents intervene more readily on errors with explicit contradictory evidence, such as looping behavior, which has a relative intervention rate of $+9.5\,pp$. In contrast, errors that remain locally coherent and require reasoning about missing actions or checks are more likely to be ignored, including premature termination and inadequate verification, which have relative intervention rates of only $-6.4\,pp$ and $-2.1\,pp$, respectively. We hope our work helps clarify the capability gaps that current agents must overcome to ultimately achieve recursive self-improvement.

Overall, our contributions are threefold:

- **A controlled benchmark for data research in the wild.** RSI-WILD uses controlled construction to recreate mixed-quality trajectory pools in the wild, enabling the evaluation of data research under fixed training and evaluation protocols.
- **An error-level analysis of data research behavior.** Using labels for 23 error types, we trace how agents intervene on failed trajectories and reveal which errors attract attention, which are often left untouched and the reason behind.
- **A systematic analysis of agents' research processes and bottlenecks.** Through controlled ablations and analyses of research traces, we examine how agents choose data strategies, use feedback, allocate their budgets, and turn external knowledge into experiments.

2 RELATED WORK

Methods for data research. Data research has progressively moved from selecting useful data to more actively transforming and creating supervision. Early work focused primarily on filtering, deduplication, quality-based selection, and data mixing to improve an existing data pool (Penedo et al., 2023; Chen et al., 2024; Liu et al., 2024a; Xie et al., 2023). Later work went beyond selection by rewriting or repairing existing examples, including refining low-quality supervision and correcting failed trajectories into useful training data (Liu et al., 2024b; Yuan et al., 2025; Fu et al., 2025). More recent work increasingly constructs new training data altogether, synthesizing tasks, executable environments, and interaction trajectories at scale (Pan et al., 2025; Xi et al., 2024; Jain et al., 2025; Yang et al., 2025).

Table 1: Comparison with representative benchmarks. ✓ indicates support, × absence, and "–" not applicable in "Main Operation" when Data Focus is ×. "Feedback / Score Isolation" indicates whether feedback and final scoring use separate data. "Runs" counts independent repetitions of the full research process.

	Research Setting			Data Research			Evaluation Protocol		
Benchmark	Data Focus	Agentic Task	Model Type	Mixed-Quality Trajectories	Failure-Aware	Operation Focus	Feedback / Score Isolation	OOD Eval	Runs
DataPrep-Bench (Liang et al., 2026)	✓	×	LLMs	×	×	Synthesis	✓	×	1
Curation-Bench (Kang et al., 2026)	✓	×	VLMs	×	×	Curation	×	×	3
RSIBench-Data (Meng et al., 2026)	✓	✓	LLMs	×	×	Synthesis	×	×	1
MLAgentBench (Huang et al., 2024)	×	×	ML	×	×	–	×	×	8
MLE-Bench (Chan et al., 2025)	×	×	ML	×	×	–	✓	×	3
MLS-Bench (Lyu et al., 2026)	×	×	ML	×	×	–	✓	✓	1
PostTrainBench (Rank et al., 2026)	×	×	LLMs	×	×	–	✓	×	3
Agent^2 RL-Bench (Chen et al., 2026b)	×	✓	LLMs	×	×	–	×	×	1
AI4AI-Bench (Chi et al., 2026)	×	×	Mixed	×	×	–	✓	×	1
RSI-Wild (ours)	✓	✓	**LLMs**	✓	✓	**Both**	✓	✓	**3**

Benchmarks for data research. Data-centric benchmarks isolate the effect of training data by fixing much of the model-training and evaluation pipeline. DataComp (Gadre et al., 2023) and DataComp-LM (Li et al., 2024) for example, fix the training recipe and ask participants to construct or select better datasets. Recent benchmarks increasingly evaluate LLMs or agents as active participants in the data-research process, rather than only evaluating static, human-designed data pipelines (Khan et al., 2025; Liang et al., 2026; Kang et al., 2026; Meng et al., 2026).

Benchmarks for recursive self-improvement. Research-agent benchmarks cover a broad range of research settings. Early benchmarks evaluate scientific research or paper reproduction (Chen et al., 2025a; Starace et al., 2025; Chen et al., 2025b; Wang et al., 2026). Other benchmarks focus more narrowly on ML engineering or method development under a specified task (Huang et al., 2024; Chan et al., 2025; Lyu et al., 2026; Chi et al., 2026). Another line of work evaluates end-to-end post-training, where agents may jointly modify data, training code, and optimization procedures (Rank et al., 2026; Chen et al., 2026b). Because these tasks involve many coupled decisions, it is often difficult to identify which research choices are responsible for the final outcome.

Table 1 compares RSI-WILD with nine representative benchmarks along these dimensions.

3 RSI-WILD

RSI-WILD evaluates AI agents' ability to conduct iterative data research under fixed training and evaluation protocols, using a public split for feedback and a disjoint hidden split for final assessment.

3.1 PROBLEM FORMULATION

Mixed-quality input. A trajectory $\tau = (q, a_1, o_1, \ldots, a_K)$ records a query q, assistant actions a_k such as reasoning steps, tool calls, and file edits, and observations o_k returned by the execution environment. The researcher receives a pool $\mathcal{P} = \mathcal{P}_{\text{succ}} \cup \mathcal{P}_{\text{fail}}$. Each failed trajectory contains an error injected at a known step, with a known type e drawn from a taxonomy $\mathcal{E}$ of 23 types. These labels are retained for analysis but are not shown to the researcher.

Iterative data research. At iteration t, the AI researcher uses the pool and its previous research history $H_{<t}$ to produce a training set $D_t = \pi(\mathcal{P}, H_{<t})$. It is told to apply any useful data operations to improve the data for the corresponding task, with access to a rollout model. When the researcher requests training, the harness applies a fixed recipe and returns public evaluation feedback.

$$M_t = \text{Train}(M_0, D_t), \qquad h_t = \text{Eval}_{\text{pub}}(M_t) \tag{1}$$

Here, M_0 is the base model, and h_t contains the public-split score, per-instance outcomes, and full rollouts. The researcher can use this feedback in its next revision, with $H_t = H_{t-1} \cup \{(D_t, h_t)\}$. Agent work, training, and public evaluation share a wall-clock budget B. Within this budget, the researcher decides when to train, when to stop, and which checkpoint $M_{t^\star}$ to submit.

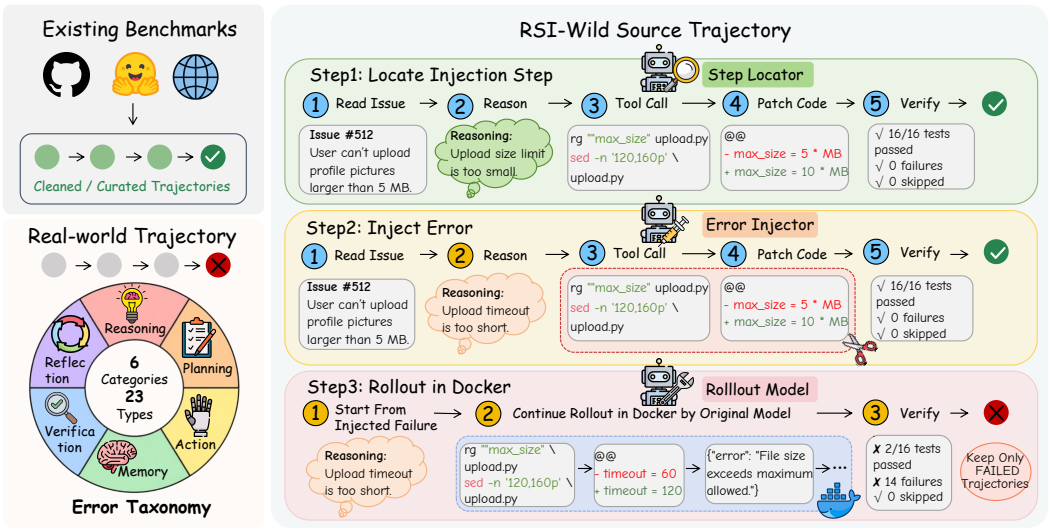

Figure 2: Trajectory pool construction of RSI-Wild.

Final evaluation. The submitted checkpoint is evaluated on a disjoint hidden split unseen by the researcher, yielding the primary score $s = \mathrm{Eval}_{\mathrm{hid}}(M_{t^\star})$.

3.2 Constructing Mixed-Quality Agentic Trajectories

Natural failed trajecotries have unknown composition, which makes error recognition unmeasurable. RSI-Wild therefore builds errors out of verified successes, as inllustrated in Figure 2.

Error taxonomy. Drawing on prior analyses of LLM agent failures (Zhu et al., 2026; Liu et al., 2026), we group 23 trajectory error types into six categories: **Planning** (4), **Reasoning** (4), **Memory** (3), **Action** (6), **Verification** (2), and **Reflection** (4). These categories capture failures at different stages of the agent workflow. Each error type further specifies the failure pattern, a plausible injection point, and the observable mistake required of the replacement step. For example, PL.4 (*Impossible Action*) targets a step where the agent chooses its next action and replaces it with a plan to perform an operation whose prerequisites are not yet satisfied. Together, these specifications determine which step to replace and what error the replacement must exhibit. Full definitions are provided in Appendix C.

Construction. Our source trajectories are successful trajectory datasets from SWE-Lego (Tao et al., 2026) for SWE-Bench (Jimenez et al., 2024) tasks and TMAX (Ivison et al., 2026) for Terminal-Bench (Merrill et al., 2026) tasks. Starting from these verified successes, our goal is to construct failed trajectories corresponding to specific error types A natural way is to prompt a model to rewrite a successful trajectory into a failing one. However, **such rewritten trajectories often look artificial and lack authentic environment feedback**. We instead intervene in an originally successful trajectory by replacing a single step with one that instantiates the target error type, and then re-roll the trajectory using its original rollout model in its original Docker environment. In this way, all subsequent actions and observations emerge naturally from the environment's response to the injected error. For each SWE and terminal task, we roll out 20 failed trajectories per error type, yielding 460 trajectories in total, and mix them with 1,040 successful ones.

The detailed procedure is as follows:

(1) Locate Injection Step: For each target error type, we use type-specific rules to score trajectory steps by their suitability for error injection, such as early exploration steps for task misunderstanding and late checking steps for inadequate verification, with details provided inAppendix C. A lightweight model (Gemini-3.5-Flash) then reranks the resulting shortlist based on how clearly each candidate matches the target subtype. Each instance is assigned at most one error type.

(2) Inject Error: We provide the injector model GPT-5.5 with the task description, the target action and its context, and error-specific instructions. It generates a replacement step that follows the original format and, unless the target error requires otherwise, preserves the step's phase and tool family to keep the injected error locally plausible.

(3) Rollout in Docker : We then resume the rollout from that step using the original rollout model in the corresponding Docker environment, so that every subsequent observation is a genuine consequence of the injected error. We only retain failed resulting trajectory.

3.3 Automated Data-Research Pipeline

Given a mixed-quality trajectory pool, the research agent runs the closed loop in Figure 1: it transforms the data, calls training, inspects public evaluation, revises, and submits a final checkpoint.

Fixed training and evaluation. Training and evaluation are exposed as callable services. Every training request uses the same recipe: full-parameter SFT of Qwen3-8B (Team, 2025b) for two epochs with a 64K-token cutoff on four A800 80GB GPUs (Details are provided in Appendix F). Each resulting checkpoint is served through the same vLLM (Kwon et al., 2023) pipeline on the same hardware, evaluated with the track-specific agent scaffold, and scored by the official verifiers.

Closed-loop data research. The researcher is free to develop data strategies such as filtering, selection, repair, rewriting, and synthesis, with Qwen3-Coder-Flash (Team, 2025b) provided as rollout model. It may invoke `train` at any time. The system trains a checkpoint, evaluates it on the public split, and returns scores, per-instance outcomes, and full rollouts as feedback for the next iteration. Research, training, and public evaluation share a fixed 25-hour budget.

Public feedback and final evaluation.

During research, feedback is available only from the public split. The final submitted checkpoint is evaluated on a disjoint hidden split that is never exposed to the researcher. This separates the feedback used for iterative development from the data used for final scoring. We additionally evaluate the submitted checkpoint on HumanEval (Chen et al., 2021), GPQA Diamond (Rein et al., 2023), AIME 2026 (Dekoninck et al., 2026), and IFBench (Pyatkin et al., 2025) to assess whether the learned data strategy transfers beyond the SWE setting or degrades general capabilities. Each researcher configuration is run independently three times. Appendix G provides the complete evaluation suite, runners, and decoding settings.

4 Experiments

4.1 Setup

Researcher agents. We evaluate 13 researcher configurations spanning 10 frontier models on their native CLI scaffolds: Claude Code (Claude Opus 5, Claude Opus 4.8, Claude Sonnet 5), Codex CLI (GPT-5.6-sol, GPT-5.6-luna, GPT-5.6-terra), and OpenCode (GLM-5.3, Kimi K3, DeepSeek V4 Pro, Gemini 3.1 Pro Preview), including a four-level reasoning-effort sweep (Low, Medium, High, Xhigh) for GPT-5.6-sol. Every configuration receives the identical prompt, pool, tooling, constraints, and 25-hour budget, and is run three times independently; we report the mean and sample standard deviation across the three runs.

Baselines. *Human* denotes a human-researcher reference, with the procedure described in Appendix E. *Train on Successful 1,500* replaces the 460 failed trajectories with their original successful counterparts while keeping the remaining 1,040 trajectories unchanged. *Train on Mixed 1,500* trains directly on the original mixed-quality pool. *Random-1,400* and *Random-1,000* train on randomly sampled subsets of the mixed 1,500-trajectory pool. *Qwen3-8B (Untrained)* denotes the base model evaluated without fine-tuning under the same protocol.

Metrics. *Public* and *Hidden* denote the resolve rates on the public and hidden splits of SWE-bench Verified. Data splits, runners, and decoding settings are fixed, as detailed in Appendix G. *Time* is the research budget consumed (h), *Traj.* the number of training trajectories, and *Avg* the unweighted mean (%) across HumanEval, GPQA Diamond, AIME 2026, and IFBench. Per-benchmark results are reported in Appendix A.5. *Intervention rate* is the fraction of the 460 labeled errors on which

the researcher intervenes, while *E–C gap* is the difference, in percentage points, between this rate and the corresponding intervention rate on the 1,040 successful trajectories.

Table 2: RSI-WILD performance on the SWE task. Frontier agents on native CLI scaffolds were evaluated with 3 independent runs (±1 standard deviation reported).

Harness	Backbone Model	SWE (↑)		Research Process		Error Intervention		General (↑)
	Model (effort)	Hidden	Public	Time (h)	Traj.	Intervention Rate (%)	E–C Gap (pp)	Avg
Codex CLI	GPT-6-astra (High)	19.3 ±2.9	15.3 ±7.0	18.5 ±1.3	1,435 ±59	20.6 ±28.6	−1.2 ±6.6	42.5 ±2.6
	GPT-5.6-sol (Medium)	19.0 ±3.6	21.0 ±3.6	18.4 ±4.1	1,395 ±118	32.7 ±56.2	−2.8 ±0.7	45.0 ±1.1
	GPT-5.6-luna (High)	18.7 ±1.5	17.3 ±2.1	18.7 ±2.3	1,389 ±149	5.5 ±9.5	−3.5 ±0.1	42.9 ±1.1
	GPT-5.6-terra (High)	17.0 ±2.6	14.7 ±4.5	16.4 ±6.2	1,473 ±23	0.1 ±0.1	−2.4 ±2.1	44.9 ±1.1
	GPT-5.6-sol (High)	16.0 ±1.0	19.7 ±2.5	16.8 ±5.3	1,439 ±42	60.6 ±52.9	+19.3 ±38.5	44.2 ±0.5
	GPT-5.6-sol (Low)	13.7 ±5.5	14.0 ±5.6	17.3 ±5.3	1,055 ±365	49.5 ±32.5	+25.9 ±11.5	41.5 ±5.1
Claude Code	Claude Sonnet 5 (High)	20.0 ±1.0	15.0 ±2.0	17.9 ±3.8	1,458 ±9	70.4 ±46.5	+2.6 ±9.3	44.1 ±0.6
	Claude Opus 4.8 (High)	17.7 ±3.1	19.0 ±2.6	18.8 ±1.6	1,440 ±33	62.5 ±29.5	+7.6 ±24.1	44.1 ±1.3
	Claude Opus 5 (High)	17.3 ±2.5	**23.0** ±1.0	21.6 ±1.2	1,486 ±25	84.7 ±24.6	+14.7 ±27.1	45.8 ±0.6
OpenCode	GLM-5.3 (Max)	**21.7** ±2.3	19.0 ±1.0	22.0 ±1.0	1,499 ±1	74.6 ±20.1	+8.4 ±36.8	45.1 ±1.4
	Kimi K3 (Max)	18.0 ±1.0	17.7 ±5.0	22.2 ±3.6	1,463 ±1	39.2 ±53.3	+3.6 ±8.9	43.7 ±3.6
	DeepSeek V4 Pro (High)	16.7 ±1.5	16.7 ±1.5	16.4 ±4.7	1,478 ±18	23.0 ±24.1	−28.8 ±29.6	42.2 ±4.7
	Gemini 3.1 Pro (High)	16.3 ±2.9	18.3 ±0.6	14.1 ±4.2	1,464 ±0	0.0 ±0.0	−3.5 ±0.0	44.2 ±0.7
–	Human	29.0	30.0	–	2,500	–	–	44.0
–	Train on Successful 1500	25.0	29.0	11.1	1,500	–	–	39.6
–	Train on mixed 1500	15.0	15.0	15.4	1,500	–	–	47.3
–	Random-1,400	16.0	10.0	17.1	1,400	–	–	45.3
–	Random-1,000	11.0	13.0	11.9	1,000	–	–	41.4
–	Qwen3-8B (Untrained)	10.0	6.0	–	–	–	–	57.6

Table 3: RSI-WILD performance on the terminal task. Due to restricted resources, each frontier agent on its native CLI scaffold has one completed run.

Harness	Backbone Model	Terminal (↑)		Research Process		Error Intervention		General (↑)
	Model (effort)	Hidden	Public	Time (h)	Traj.	Intervention Rate (%)	E–C Gap (pp)	Avg
Codex CLI	GPT-6-astra (High)	6.0	**14.0**	8.7	1,310	19.3	+9.6	20.3
	GPT-5.6-sol (High)	6.0	10.0	2.8	1,354	74.6	+15.8	21.4
	GPT-5.6-terra (High)	**10.0**	12.0	10.1	1,500	85.2	+56.9	22.2
Claude Code	Claude Sonnet 5 (High)	**10.0**	10.0	4.2	589	92.0	+45.0	18.6
	Claude Opus 5 (High)	6.0	12.0	12.0	1,278	25.0	+13.5	17.1
OpenCode	GLM-5.3 (Max)	**10.0**	12.0	23.5	1,379	100.0	0.0	**25.5**
	Kimi K3 (Max)	**10.0**	10.0	15.9	712	85.9	+43.3	17.6
–	Train on Successful 1500	14.0	14.0	–	1,500	–	–	52.5
–	Train on mixed 1500	6.0	6.0	–	1,500	–	–	19.8
–	Qwen3-8B (Untrained)	2.0	2.0	–	0	–	–	56.2

4.2 MAIN RESULTS

Table 2 reports the performance of all researcher configurations. We make the following three observations. **(1) Research agents recover substantial value from noisy data.** Twelve of the 13 configurations outperform training directly on the mixed pool on the hidden split, and all outperform the untrained model. GLM-5.3, for example, reaches $21.7 \pm 2.3\%$, compared with 15.0% for the mixed- pool baseline and 10.0% for the untrained model. Most configurations also outperform their size-matched random-selection baselines, further confirming that the gains come from effective data strategies. **(2) A substantial gap remains.** The best agent score, 21.7%, remains below training on 1,500 successful trajectories (25.0%) and the human researcher (29.0%), indicating room for

improvement. **(3) Public feedback may misguide checkpoint selection.** Claude Opus 5 achieves the highest mean public score (23.0%) but only 17.3% on the hidden split. Trace analysis shows that across all three runs, it makes substantial revisions in later iterations, yet ultimately selects the first-iteration checkpoint as its final submission based on public feedback. This suggests that public feedback can strongly shape model selection without necessarily identifying the checkpoint that generalizes best to the hidden split. **(4) Results on terminal tasks align with the findings above.** We evaluate seven researcher configurations on TBLite with 50 public and 50 hidden tasks, as shown in Table 3. Detailed results on the terminal task are reported in Appendix B.

4.3 Feedback-Driven Iterative Research

Current agents do not improve reliably through this feedback loop. RSI-Wild allows researchers to observe public evaluation results after each training run, revise their data strategies, and retrain. However, current agents do not improve reliably through this feedback loop. In 22 of these 34 runs, the second attempt scores lower on the public split than the first. Thus, reacting to feedback often leads to regressions rather than steady progress.

Long-horizon iteration shows gradual gains. To study longer-term behavior, we conduct a special long-horizon run in which the researcher (GPT-5.6-sol Max) receives only half of the trajectory pool. Under the same 25-hour budget, this setting allows 11 valid training attempts. Figure 7 shows that although the public score continues to fluctuate substantially across iterations, it shows a gradual upward trend overall, with a fitted slope of approximately +0.37 points per iteration.

4.4 Budget Study

Current agents are poor budget allocators. Although we explicitly informs researchers that training is expensive and encourages substantial investigation before launching a training run, current agents still tend to enter the train-and-evaluate loop quickly. As shown in Figure 8, across configurations, the median researcher spends only about 3% of the total runtime on active research, while the remaining 97% is spent waiting for training and evaluation. This suggests that agents still allocate limited compute poorly, often launching expensive training runs before sufficiently analyzing the data or developing their strategies.

Agents often stop early, while using more budget does not reliably improve performance. As illustrated in Figure 3, many agents terminate their research and submit a final result while substantial budget remains unused. For example, Gemini 3.1 Pro stops after roughly 14 hours, leaving about 11 hours unused. At the same time, greater budget consumption does not reliably translate into better performance: the correlation between budget consumed and final performance is weak and statistically insignificant. The bottleneck therefore appears to be not simply insufficient time or compute, but the agents' limited ability to decide when to continue researching, when to train, and when the available evidence justifies another iteration.

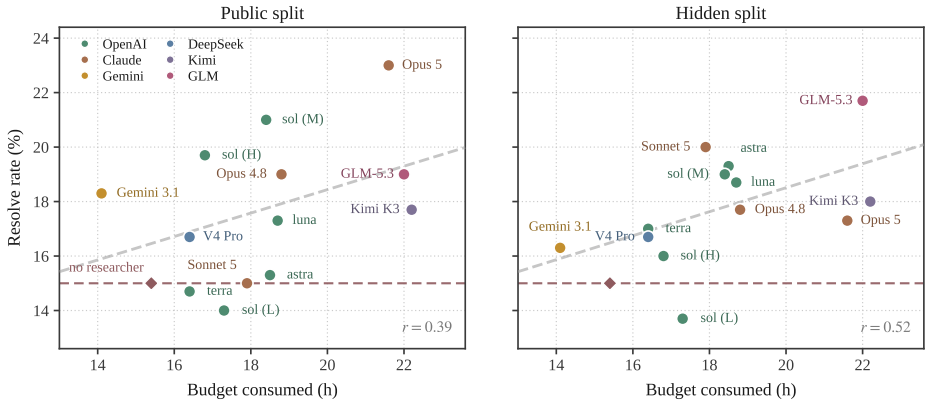

Figure 3: Budget consumed versus public and hidden resolve rates on the SWE task. Each point represents the mean over three runs; r denotes the Pearson correlation across configurations.

5 Ablation and Analysis

We next examine the mechanisms behind these outcomes by asking three questions.

5.1 How Capable Are Agents at Different Data Operations?

To measure the researcher's atomic capabilities across different data operations, we fix the researcher to GPT-5.6-sol High and compare four settings under otherwise identical conditions on the SWE task: Repair-only, Synthesis-only, and Filtering-only, where the researcher is restricted to a single operation, and an unrestricted setting in which it can freely choose among data operations.

Repair performs best, while synthesis is shallow and filtering is overly coarse. Repair-only achieves the highest hidden score of 20.3, outperforming Synthesis-only at 16.3, Auto at 16.0, and Filtering-only at 14.7, as reported in Table 4. Synthesis expands the dataset to about 1,769 trajectories, but many generated trajectories are shallow and repetitive. Filtering, meanwhile, relies on coarse trajectory-level signals and can remove useful supervision together with explicit errors. Repair is more effective because it makes localized, evidence-backed corrections while preserving the useful parts of the original trajectories.

Repair works best on locally verifiable errors, while semantic errors remain challenging. Trace analysis shows that its gains mainly come from addressing defects that can be identified from explicit contextual or tool feedback, such as non-zero exit codes, parsing errors, and repeated tool calls. In contrast, semantic errors such as goal drift, false memory, reflective hallucination, and planning–action mismatches remain substantially harder to detect and correct.

Table 4: Data-operation ablation.

Permitted operations	Hidden (%)	Public (%)	Time (h)	Traj.	Gen. (%)
Auto (unrestricted)	16.0 ±1.0	19.7 ±2.5	16.8 ±5.3	1,439 ±42	44.2
Repair only	**20.3** ±0.6	16.7 ±2.1	23.5 ±2.1[a]	1,500 ±0	44.7
Synthesis only	16.3 ±1.5	17.3 ±2.1	13.7 ±3.8	1,769 ±208	44.9
Filter only	14.7 ±2.3	11.0 ±2.0	16.0 ±1.4	1,015 ±235	41.2
No researcher	15.0	15.0	15.4	1,500	47.3

5.2 Does More External Knowledge Improve Performance?

One possible bottleneck is that researchers lack knowledge of data-research methods. To test this, we fix the researcher to GPT-5.6-sol High on the SWE task, provide the same set of 30 data-research methods in three forms: GitHub repositories, research papers, and prepared skills.

Easier access increases method consumption but brings limited performance gains. Agents read more methods as resources become easier to consume, increasing from 6.0 for repositories to 12.0 for papers and 16.7 for skills, as reported in Table 5. However, this increase does not translate into consistent performance gains: hidden scores are 17.0%, 19.7%, and 17.3%, respectively, compared with 16.0% without external resources.

A knowledge–action gap limits the value of external resources. Trace analysis shows that agents can locate and explain relevant methods, but often reduce them to simplified heuristics. For example, they may reduce a trajectory filtering method to checking for a termination string or replacing learned quality scores with keyword counts. Thus, the main bottleneck is not access to methodological knowledge, but the ability to translate that knowledge into concrete experiments.

5.3 Which Errors Do Research Agents Prefer to Intervene On?

We examine the SWE task for which failed trajectories research agents are more likely to intervene. A trajectory is counted as intervened on if it is removed, rewritten, or its `turn_mask` is semantically changed. Figure 4 summarizes intervention patterns across researcher configurations and error types. Full numerical results are provided in Appendix D.

Table 5: Reference-resource ablation.

Resource given	Methods Read	Hidden (%)	Public (%)	Time (h)	Traj.	Gen. (%)
None	–	16.0 ±1.0	**19.7** ±2.5	16.8 ±5.3	1,439 ±42	44.2
GitHub repository	6.0	17.0 ±1.0	14.3 ±2.1	9.0 ±1.5	1,337 ±224	44.7
Research paper	12.3	**19.7** ±3.2	13.3 ±2.1	10.9 ±2.0	1,405 ±135	**45.1**
Skill file	16.7	17.3 ±1.1	15.0 ±7.6	16.3 ±7.3	1,296 ±257	44.3
No researcher	–	15.0	15.0	15.4	1,500	47.3

Different research agents exhibit distinct intervention preferences across error types. Figure 4 shows that researchers differ not only in overall intervention rates and operations, but also in which error types they preferentially target after normalizing for each run's average intervention tendency. Table 11 further examines whether these preferences persist across runs. Some are consistent: GPT-5.6-sol (Low) intervenes above the correct-trajectory baseline on 16 error types in all three runs, whereas GPT-5.6-luna (High) and DeepSeek V4 Pro (High) remain below baseline on 17 types. Other configurations show little agreement.

Agents intervene more often on errors with explicit contradictory evidence. As shown in Table 14, looping behavior receives the highest relative intervention rate, at $+9.5$ pp over correct trajectories. By contrast, premature termination and inadequate verification are intervened on less often, with relative rates of -6.4 pp and -2.1 pp, respectively. This suggests that agents are more likely to detect errors that conflict with evidence already present in the trajectory, while failures that remain locally plausible and require inferring missing actions or checks are more easily overlooked.



Figure 4: Researcher interventions on the 460 labeled failures. **Left:** intervention rate for each error type relative to the run's mean over all failures, averaged across three runs. **Right:** final status of the failures; red ticks denote the intervention rate on the 1,040 correct trajectories. Ticks to the left of the intervention boundary indicate higher intervention rates on failures than on correct trajectories.

6 Conclusion

We introduce RSI-Wild, a benchmark for evaluating data research agents on mixed-quality agentic trajectory pools. It combines controlled, execution-grounded errors with unrestricted data operations and fixed training and evaluation, enabling auditable assessment of agents' data strategies. Current agents recover value from imperfect data but still trail human researchers and exhibit unstable feedback-driven improvement. Our analyses further reveal AI researchers' weaknesses in budget allocation, error diagnosis, and translating external knowledge into concrete experiments. These findings highlight that executing data operations is not equivalent to conducting reliable research. RSI-Wild provides a testbed for developing agents that better diagnose data problems, select appropriate interventions, and validate improvements toward recursive self-improvement.

AI USE STATEMENT

In this work, we used generative AI tools to assist with and polish the writing of the paper. We also used generative AI models as components of our benchmark construction pipeline, including locating candidate steps for error injection, generating error-injected actions, and re-rolling out trajectories in the original execution environments. These uses are described in the corresponding methodology sections of the paper. We reviewed all AI-assisted writing and verified the generated trajectories through the benchmark construction and execution-based validation procedures described in the paper. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the source code and experiment scripts as part of the supplementary materials. Detailed descriptions of the benchmark construction, trajectory generation and error injection procedures, training configurations, evaluation protocols, and implementation details are provided in the main paper and appendix.

REFERENCES

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. Deepspeed-inference: Enabling efficient inference of transformer models at unprecedented scale. In Felix Wolf, Sameer Shende, Candace Culhane, Sadaf R. Alam, and Heike Jagode (eds.), *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, November 13-18, 2022*, pp. 46:1–46:15. IEEE, 2022. doi: 10.1109/SC41404.2022.00051. URL https://doi.org/10.1109/SC41404.2022.00051.

Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, Zeyao Ma, Kashun Shum, Xuwu Wang, Jinxi Wei, Jiaxi Yang, Jiajun Zhang, Lei Zhang, Zongmeng Zhang, Wenting Zhao, and Fan Zhou. Qwen3-coder-next technical report. *CoRR*, abs/2603.00729, 2026. doi: 10.48550/ARXIV.2603.00729. URL https://doi.org/10.48550/arXiv.2603.00729.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. Mlebench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. Mlr-bench: Evaluating AI agents on open-ended machine learning research. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025a. URL http://papers.nips.cc/paper_files/paper/2025/hash/ab8dd000d6f87f40061a73f8bca7fae4-Abstract-Datasets_and_Benchmarks_Track.html.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=FdVXgSJhvz.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan,

Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in AI: from bounded self-refinement to autonomous research loops. *CoRR*, abs/2607.07663, 2026a. doi: 10.48550/ARXIV.2607.07663. URL https://doi.org/10.48550/arXiv.2607.07663.

Wanyi Chen, Xiao Yang, Xu Yang, Tianming Sha, Qizheng Li, Zhuo Wang, Bowen Xian, Fang Kong, Weiqing Liu, and Jiang Bian. Agentˆ2 rl-bench: Can LLM agents engineer agentic RL post-training? *CoRR*, abs/2604.10547, 2026b. doi: 10.48550/ARXIV.2604.10547. URL https://doi.org/10.48550/arXiv.2604.10547.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=6z4YKr0GK6.

Yizhe Chi, Wenyi Li, Deyao Hong, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. Ai4ai-bench: Benchmarking LLM agents in algorithmic design for recursive self-improvement. *CoRR*, abs/2608.20318, 2026. doi: 10.48550/ARXIV.2608.20318. URL https://doi.org/10.48550/arXiv.2608.20318.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. Agentrefine: Enhancing agent generalization through refinement tuning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=FDimWzmcWn.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah M. Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander J. Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/56332d41d55ad7ad8024aac625881be7-Abstract-Datasets_and_Benchmarks.html.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller,

Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *CoRR*, abs/2606.23321, 2026. doi: 10.48550/ARXIV.2606.23321. URL https://doi.org/10.48550/arXiv.2606.23321.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights SWE agents. *CoRR*, abs/2504.07164, 2025. doi: 10.48550/ARXIV.2504.07164. URL https://doi.org/10.48550/arXiv.2504.07164.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Feiyang Kang, Hanze Li, Adam Nguyen, Mahavir Dabas, Jiaqi W. Ma, Frederic Sala, Dawn Song, and Ruoxi Jia. Can generalist agents automate data curation? *CoRR*, abs/2606.04261, 2026. doi: 10.48550/ARXIV.2606.04261. URL https://doi.org/10.48550/arXiv.2606.04261.

Zaid Khan, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. Dataenvgym: Data generation agents in teacher environments with student feedback. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=00SnKBGTsz.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Scott Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F. Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah M. Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Raghavi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alex Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/19e4ea30dded58259665db375885e412-Abstract-Datasets_and_Benchmarks_Track.html.

Hao Liang, Qifeng Cai, Yibo Lin, Jianzhuo Du, Qifeng Xia, Sizhe Qiu, Linzhuang Sun, Meiyi Qiang, Zhaoyang Han, Xiaochen Ma, Bohan Zeng, Ruichuan An, Conghui He, and Wentao Zhang. Dataprep-bench: Benchmarking llms as training data preparators. *CoRR*, abs/2607.20465, 2026. doi: 10.48550/ARXIV.2607.20465. URL https://doi.org/10.48550/arXiv.2607.20465.

Jiale Liu, Huajun Xi, Shaokun Zhang, Yifan Zeng, Tianwei Yue, Chi Wang, Jian Kang, Qingyun Wu, and Huazheng Wang. Who&when pro: Can llms really attribute failures in AI agents? *CoRR*, abs/2607.09996, 2026. doi: 10.48550/ARXIV.2607.09996. URL https://doi.org/10.48550/arXiv.2607.09996.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? A comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=BTKAeLqLMw.

Yilun Liu, Shimin Tao, Xiaofeng Zhao, Ming Zhu, Wenbing Ma, Junhao Zhu, Chang Su, Yutai Hou, Miao Zhang, Min Zhang, Hongxia Ma, Li Zhang, Hao Yang, and Yanfei Jiang. Coachlm: Automatic instruction revisions improve the data quality in LLM instruction tuning. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*, pp. 5184–5197. IEEE, 2024b. doi: 10.1109/ICDE60146.2024.00390. URL https://doi.org/10.1109/ICDE60146.2024.00390.

Bohan Lyu, Yucheng Yang, Siqiao Huang, Jiaru Zhang, Qixin Xu, Xinghan Li, Xinyang Han, Yicheng Zhang, Huaqing Zhang, Runhan Huang, Kaicheng Yang, Zitao Chen, Wentao Guo, Junlin Yang, Xinyue Ai, Wenhao Chai, Yadi Cao, Ziran Yang, Kun Wang, Dapeng Jiang, Huan-ang Gao, Shange Tang, Chengshuai Shi, Simon S. Du, Max Simchowitz, Jiantao Jiao, Dawn Song, and Chi Jin. Mls-bench: A holistic and rigorous assessment of AI systems on building better AI. *CoRR*, abs/2605.08678, 2026. doi: 10.48550/ARXIV.2605.08678. URL https://doi.org/10.48550/arXiv.2605.08678.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. Rsibench-data: Benchmarking data-centric research for recursive self-improvement. *CoRR*, abs/2607.25886, 2026. doi: 10.48550/ARXIV.2607.25886. URL https://doi.org/10.48550/arXiv.2607.25886.

Mike A. Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, Estefany Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis-Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *CoRR*, abs/2601.11868, 2026. doi: 10.48550/ARXIV.2601.11868. URL https://doi.org/10.48550/arXiv.2601.11868.

Bespoke Labs OpenThoughts-Agent team, Snorkel AI. OpenThoughts-TBLite: A High-Signal Benchmark for Iterating on Terminal Agents. https://www.openthoughts.ai/blog/openthoughts-tblite, February 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data only. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/fa3ed726cc5073b9c31e3e49a807789c-Abstract-Datasets_and_Benchmarks.html.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/46499a0622ecf568b72d17b61e45dbd5-Abstract-Datasets_and_Benchmarks_Track.html.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. Posttrainbench: Can LLM agents automate LLM post-training? *CoRR*, abs/2603.08640, 2026. doi: 10.48550/ARXIV.2603.08640. URL https://doi.org/10.48550/arXiv.2603.08640.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate AI research. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

Chaofan Tao, Jierun Chen, Yuxin Jiang, Kaiqi Kou, Shaowei Wang, Ruoyu Wang, Xiaohui Li, Sidi Yang, Yiming Du, Jianbo Dai, Zhiming Mao, Xinyu Wang, Lifeng Shang, and Haoli Bai. Swe-lego: Pushing the limits of supervised fine-tuning for software issue resolving. *CoRR*, abs/2601.01426, 2026. doi: 10.48550/ARXIV.2601.01426. URL https://doi.org/10.48550/arXiv.2601.01426.

Kimi Team. Kimi K2: open agentic intelligence. *CoRR*, abs/2507.20534, 2025a. doi: 10.48550/ARXIV.2507.20534. URL https://doi.org/10.48550/arXiv.2507.20534.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025b. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*,

volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

Yuru Wang, Lejun Cheng, Yuxin Zuo, Sihang Zeng, Bingxiang He, Che Jiang, Junlin Yang, Yuchong Wang, Kaikai Zhao, Weifeng Huang, Kai Tian, Zhenzhao Yuan, Jincheng Zhong, Weizhi Wang, Ning Ding, Bowen Zhou, and Kaiyan Zhang. Naturebench: Can coding agents match the published SOTA of nature-family papers? *CoRR*, abs/2606.24530, 2026. doi: 10.48550/ARXIV.2606.24530. URL https://doi.org/10.48550/arXiv.2606.24530.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Agentgym: Evolving large language model-based agents across diverse environments. *CoRR*, abs/2406.04151, 2024. doi: 10.48550/ARXIV.2406.04151. URL https://doi.org/10.48550/arXiv.2406.04151.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/dcba6be91359358c2355cd920da3fcbd-Abstract-Conference.html.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/8b86cf5ace600c48fd188efbb8dedec8-Abstract-Datasets_and_Benchmarks_Track.html.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *CoRR*, abs/2501.11425, 2025. doi: 10.48550/ARXIV.2501.11425. URL https://doi.org/10.48550/arXiv.2501.11425.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 400–410. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-DEMOS.38. URL https://doi.org/10.18653/v1/2024.acl-demos.38.

Kunlun Zhu, Xuyan Ye, Zhiguang Han, Yuchen Zhao, Bingxuan Li, Weijia Zhang, Muxin Tian, Xiangru Tang, Pan Lu, James Zou, Jiaxuan You, and Heng Ji. Agentdebugx: An open-source toolkit for failure observability, attribution, and recovery in LLM agents. *CoRR*, abs/2607.18754, 2026. doi: 10.48550/ARXIV.2607.18754. URL https://doi.org/10.48550/arXiv.2607.18754.

A Additional SWE Results

This appendix provides additional results and analysis for the SWE task.

A.1 Hidden-Score Ranking and Size-Matched References

Figure 5 ranks the standard researcher configurations by mean hidden-split score and shows the size of each submitted dataset.

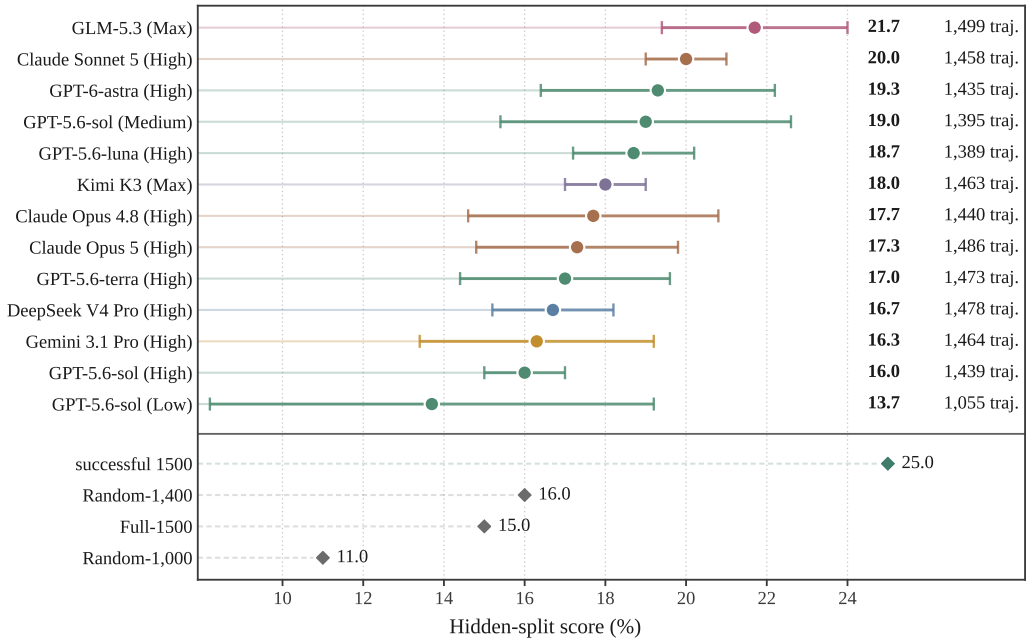

Figure 5: Hidden-split score of every standard configuration on the SWE task, ranked, with the size of the submitted dataset alongside. The dot is the 3-run mean and the whisker spans $\pm$ 1 sample standard deviation; point colour encodes model family. Below the rule are the no-researcher reference points, each a single seed: the aligned clean twin of the same 1,500 instances with no injected errors, uniform random 1,400-row and 1,000-row subsets of the noisy pool (seed 42), and the full 1,500-row noisy pool trained directly.

A.2 Public Scores Across Training Attempts

Figure 6 shows the public score after each valid training attempt, both for individual runs and as a configuration mean.

Figure 7 shows the longer GPT-5.6-sol Max run, which uses half of the trajectory pool (750 examples) as input. The smaller training set allows 11 valid attempts within the same 25-hour budget. Scores fluctuate between attempts but have a fitted upward trend of about 0.37 public-score points per attempt.

A.3 Research Time and Training Time

Figure 8 separates each run's active data-research intervals from the training and public-evaluation cycles requested by the agent. Across configurations, the median researcher spends about 3% of its runtime on active research; training and evaluation account for the remaining 97%. The plot also shows when runs submit early, leaving part of the 25-hour budget unused.

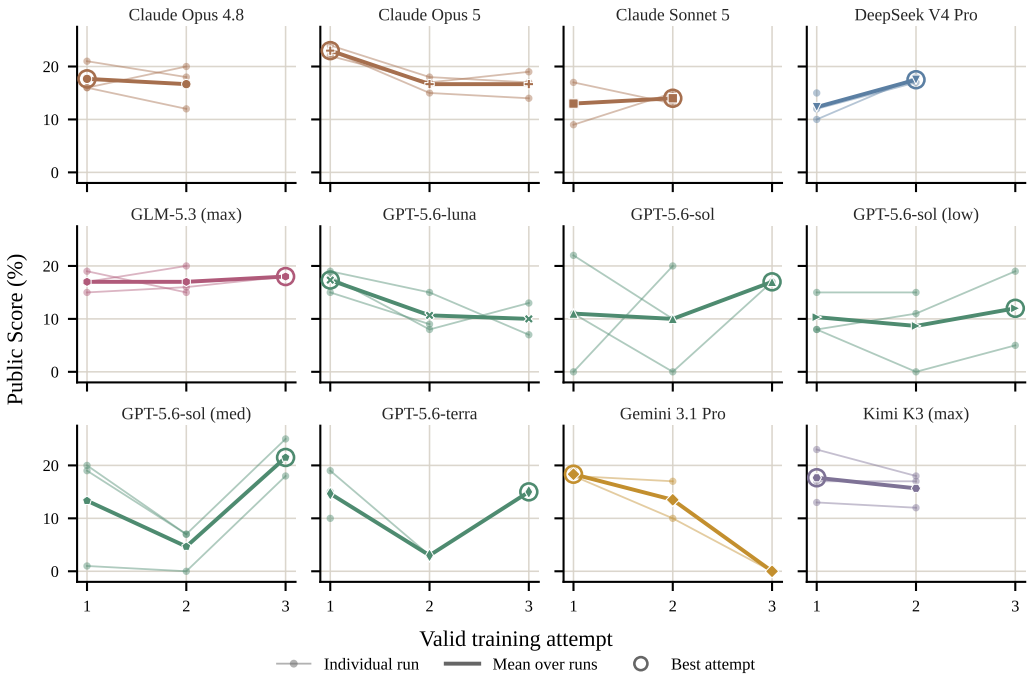

Figure 6: Public score at each valid training attempt on the SWE task. Thin lines are the individual runs of a configuration and the thick line is their mean; a circle marks the attempt at which the mean reaches its best.

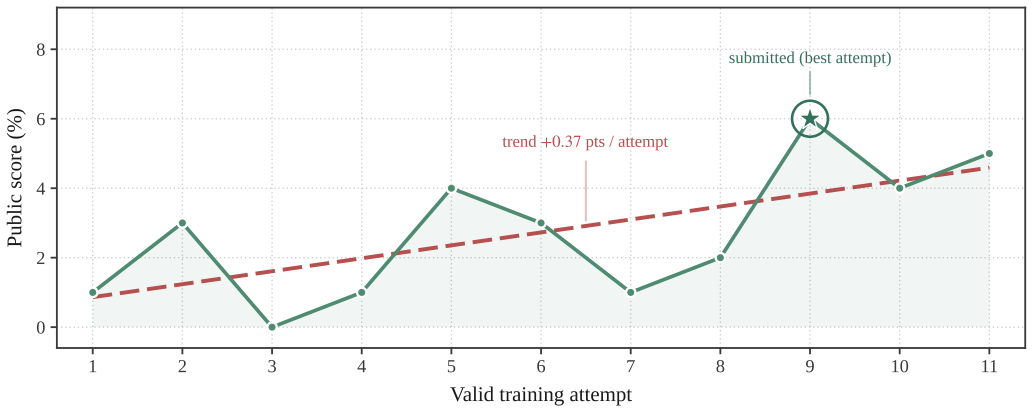

Figure 7: Public score across valid training attempts for GPT-5.6-sol Max on the SWE task, using half of the trajectory pool (750 trajectories) as input to enable more iterations.



Figure 8: 25-Hour Budget Usage for SWE Task Runs. Each row represents one run, grouped by configuration and ordered by configuration-level mean hidden score. Hatched green intervals show periods of active agent work on data research while gray intervals show training and public evaluation. The red dashed line marks the budget limit.

A.4 Scaling Researcher Capability

We test whether more reasoning effort or a larger researcher model reliably improves the final data strategy. In the reasoning-effort ablation, we hold the researcher at GPT-5.6-sol on Codex CLI and vary only its effort level (Low, Medium, or High). In the model-tier ablation, we compare GPT-5.6-terra, luna, and sol at High effort on the same CLI. Both comparisons use the standard prompt, mixed-quality pool, 25-hour budget, and fixed training and evaluation services; each configuration has three independent runs. Hidden-split resolve rate is the primary outcome.

Neither comparison is monotonic. As Table 6 shows, Medium effort achieves the highest hidden score within the sol sweep (19.0), compared with 13.7 at Low and 16.0 at High. Across model tiers, Table 7 shows that luna achieves the highest hidden score (18.7), above terra (17.0) and sol (16.0), although sol attains the highest public score. These limited comparisons do not imply that additional model capability or reasoning is generally harmful. They show that scaling researcher capability alone does not reliably scale research outcome in this setting.

Table 6: Reasoning-effort ablation for GPT-5.6-sol on Codex CLI (3 runs per level).

Effort	Hidden (%)	Public (%)	Time (h)	Traj.	Gen. (%)
Low	13.7 ±5.5	14.0 ±5.6	17.3 ±5.3	1,055 ±365	41.5
Medium	**19.0** ±3.6	**21.0** ±3.6	18.4 ±4.1	1,395 ±118	45.0
High	16.0 ±1.0	19.7 ±2.5	16.8 ±5.3	1,439 ±42	44.2
No researcher	15.0	15.0	15.4	1,500	47.3

Table 7: Researcher model-tier ablation within the GPT-5.6 family (Codex CLI scaffold, high effort, standard prompt), 3 runs per model.

GPT-5.6 tier	Hidden (%)	Public (%)	Time (h)	Traj.	Gen. (%)
terra	17.0 ±2.6	14.7 ±4.5	16.4 ±6.2	1,473 ±23	44.9
luna	**18.7** ±1.5	17.3 ±2.1	18.7 ±2.3	1,389 ±149	43.0
sol	16.0 ±1.0	**19.7** ±2.5	16.8 ±5.3	1,439 ±42	44.2
No researcher	15.0	15.0	15.4	1,500	47.3

A.5 General Benchmark Results

Table 8 gives the four general-benchmark scores summarized by the Avg column of Table 2. The average is the unweighted mean of HumanEval, GPQA Diamond, AIME 2026, and IFBench. Each researcher configuration reports the mean and sample standard deviation across three runs.

Table 8: General-benchmark scores (%) for the configurations in Table 2. Values for researcher configurations are means with ±1 sample standard deviation across three runs. Avg is computed per run as the unweighted mean of the four benchmark scores, then averaged across runs.

Harness	Backbone Model (effort)	HumanEval	GPQA Diamond	AIME 2026	IFBench	Avg
Codex CLI	GPT-6-astra (High)	68.9 ±9.2	36.7 ±3.9	39.7 ±2.5	24.6 ±0.8	42.5 ±2.6
	GPT-5.6-sol (Medium)	78.0 ±1.8	37.9 ±1.0	**38.9** ±2.1	**25.3** ±1.7	45.0 ±1.1
	GPT-5.6-luna (High)	74.0 ±5.9	38.6 ±2.3	35.6 ±2.5	23.7 ±1.5	42.9 ±1.1
	GPT-5.6-terra (High)	**80.7** ±1.9	37.4 ±3.5	37.2 ±4.6	24.3 ±0.6	44.9 ±1.1
	GPT-5.6-sol (High)	76.2 ±4.0	39.6 ±3.4	38.3 ±1.4	22.8 ±0.2	44.2 ±0.5
	GPT-5.6-sol (Low)	66.1 ±15.9	39.7 ±5.3	38.3 ±1.7	21.8 ±1.3	41.5 ±5.1
Claude Code	Claude Sonnet 5 (High)	79.3 ±2.2	38.0 ±1.8	35.3 ±1.7	23.9 ±2.2	44.1 ±0.6
	Claude Opus 4.8 (High)	75.8 ±7.6	41.4 ±4.6	36.9 ±2.7	22.1 ±1.6	44.1 ±1.3
	Claude Opus 5 (High)	79.5 ±3.7	41.2 ±1.6	38.6 ±1.0	24.0 ±0.3	45.8 ±0.6
OpenCode	GLM-5.3 (Max)	78.9 ±3.1	**41.4** ±2.8	37.5 ±2.2	22.6 ±1.6	45.1 ±1.4
	Kimi K3 (Max)	74.2 ±7.7	39.4 ±2.8	37.5 ±4.3	23.6 ±4.7	43.7 ±3.6
	DeepSeek V4 Pro (High)	71.8 ±15.2	39.6 ±3.8	35.6 ±4.1	21.9 ±2.5	42.2 ±4.7
	Gemini 3.1 Pro Preview (High)	74.2 ±2.5	40.9 ±3.0	38.9 ±1.9	22.8 ±0.5	44.2 ±0.7
–	Human	61.0	50.0	40.0	25.0	44.0
–	Train on Successful 1500	46.3	48.0	41.7	22.3	39.6
–	Train on mixed 1500	79.3	40.4	44.2	25.3	47.3
–	Random-1,400	81.7	36.9	39.2	23.3	45.3
–	Random-1,000	78.0	32.8	33.3	21.3	41.4
–	Qwen3-8B (Untrained)	76.8	56.6	61.7	35.3	57.6

B Terminal-Bench Results

B.1 Terminal-Track General Benchmark Results

Table 9 reports the four general-benchmark scores underlying the Avg column of Table 3. Each researcher configuration has one run, and each reference has one evaluation, so these are individual scores rather than means with standard deviations. As in Table 2, Avg is the unweighted mean of HumanEval, GPQA Diamond, AIME 2026, and IFBench.

Table 9: General-benchmark scores (%) for the terminal track (one completed run per researcher configuration and one evaluation per reference). Avg is the mean of the four benchmark scores.

Harness	Backbone Model (effort)	HumanEval	GPQA Diamond	AIME 2026	IFBench	Avg
Codex CLI	GPT-6-astra (High)	10.37	**27.78**	**27.50**	15.67	20.33
	GPT-5.6-sol (High)	28.05	27.27	16.67	13.67	21.42
	GPT-5.6-terra (High)	31.71	24.75	19.17	13.33	22.24
Claude Code	Claude Sonnet 5 (High)	20.12	22.22	14.17	**18.00**	18.63
	Claude Opus 5 (High)	18.29	**27.78**	12.50	10.00	17.14
OpenCode	GLM-5.3 (Max)	**46.95**	26.26	14.17	14.67	**25.51**
	Kimi K3 (Max)	28.05	16.67	12.50	13.33	17.64
–	Train on Successful 1500	85.37	47.47	47.50	29.67	52.50
–	Train on mixed 1500	23.78	18.69	21.67	15.00	19.79
–	Original Qwen3-8B (Untrained)	77.44	53.03	60.83	33.33	56.16

B.2 Terminal-Track Research Dynamics

Figures 9–11 show the terminal-track counterparts of Figures 3, 6, and 8. Each of the seven configurations has one completed run, so no cross-run averaging is applied.

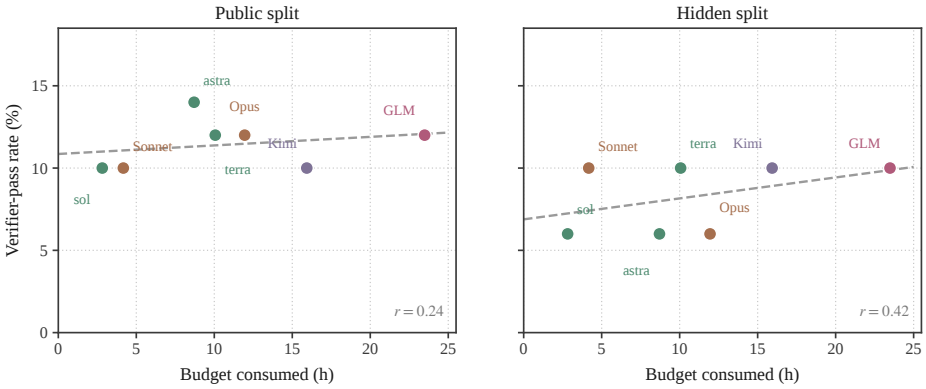

Figure 9: Budget consumed versus public and hidden verifier-pass rates on the terminal track. Each point is one completed run. Public and Hidden is the public and hidden evaluation of the submitted checkpoint. The r values are Pearson correlations across the seven runs.

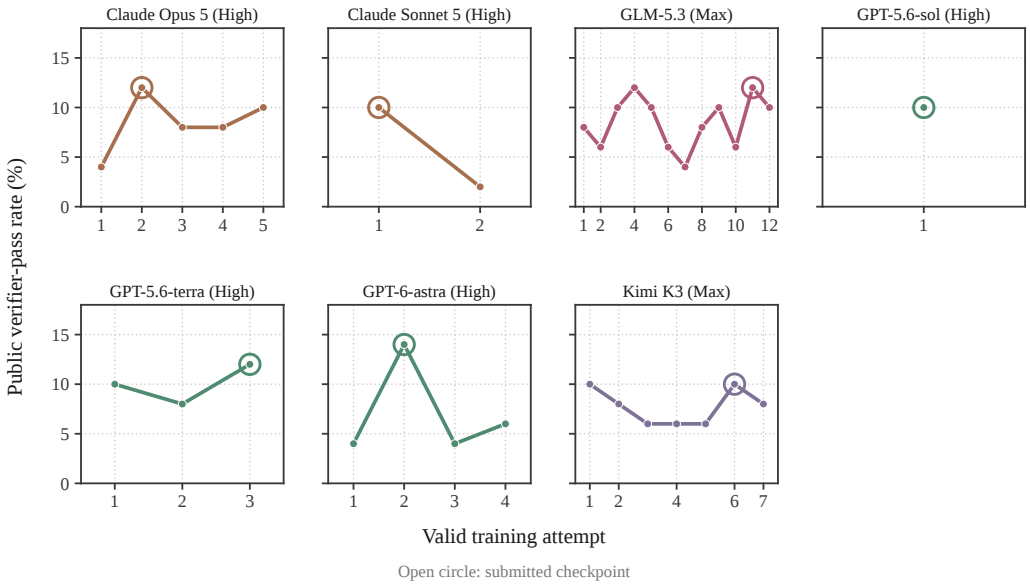

Figure 10: Public verifier-pass rate at each valid training attempt on the terminal track. Failed dataset validation attempts are excluded. Each panel shows one run; an open circle marks the checkpoint the agent submitted. All rates divide resolved tasks by 50; failed or missing trials count as incorrect.



Figure 11: 25-Hour Budget Usage for Terminal Task Runs. Each row represents one run, grouped by configuration and ordered by configuration-level mean hidden score. Hatched green intervals show periods of active agent work on data research while gray intervals show training and public evaluation. The red dashed line marks the budget limit.

C Error Taxonomy

This appendix gives the full specification of the 23 trajectory error types used to construct RSI-Wild. Table 10 lists every type with its definition for quick reference, and Sections C.1–C.6 give the complete per-type specification. The code prefixes identify the six categories: planning (PL), reasoning (R), memory (M), action (A), verification (V), and reflection (RF).

Each type is specified by three fields that jointly determine how it is realized in a trajectory. The *definition* states the error mode. The *injection point* states where in a trajectory the error characteristically occurs, and is what the type-specific selection rules of Step 1 (Section 3.2) score candidate steps against. The *injected behavior* states what the rewritten step must do, and is what the injector model of Step 2 is instructed to produce. Every replacement step must remain a single valid action in the CodeAct action space used by the source rollouts, namely `think` for reasoning steps, `execute_bash` for shell commands, `str_replace_editor` for file viewing and editing, and `finish` for termination; each type additionally declares which of these actions are plausible carriers for it, so that an injected error stays within the phase and tool family of the step it replaces.

C.1 Planning Errors

Planning errors corrupt the strategy a trajectory commits to, typically before or between the main edits, so their consequences surface many steps later.

- **PL.1 Ineffective Planning.** Forms an unsound or inefficient high-level strategy. *Injection point:* early planning, exploration, or task decomposition before the main edits. *Injected behavior:* choose a strategy that sounds reasonable but is poorly decomposed or rests on a weak premise.
- **PL.2 Goal Drift.** Gradually deviates from the original objective during multi-step execution. *Injection point:* middle or later reasoning after partial progress has been made. *Injected behavior:* shift attention to a tangential cleanup, refactor, or adjacent concern that does not resolve the issue.
- **PL.3 Constraint Ignorance.** Ignores explicit or implicit constraints such as time, budget, dependencies, or environment restrictions. *Injection point:* a decision point where tests,

Table 10: The 23 trajectory error types of RSI-WILD.

Code	Name	Definition
Planning		
PL.1	Ineffective Planning	Forms an unsound or inefficient high-level strategy.
PL.2	Goal Drift	Gradually deviates from the original objective during multi-step execution.
PL.3	Constraint Ignorance	Ignores explicit or implicit constraints such as dependencies or environment limits.
PL.4	Impossible Action	Plans actions whose preconditions are not satisfied in the current state.
Reasoning		
R.1	Hallucination	Makes claims not grounded in retrieved observations, tool outputs, or evidence.
R.2	Reasoning Error	Misapplies correct information via entity confusion, faulty inference, or missed edge cases.
R.3	Calculation Error	Arithmetic, counting, off-by-one, unit, or rounding mistakes.
R.4	Task Misunderstanding	Misinterprets the user's intent or the task scope.
Memory		
M.1	Retrieval Failure	Fails to retrieve relevant information from history or previous observations.
M.2	False Memory	Incorrectly recalls facts, history, or prior states that never occurred.
M.3	Incomplete Summary	Compresses stored information too aggressively, dropping details that later matter.
Action		
A.1	Tool / Parameter Error	Uses the correct tool with incorrect, malformed, or missing parameters.
A.2	Hallucinated Tool	Invokes non-existent tools, APIs, or actions unavailable in the environment.
A.3	Output Format Error	Produces malformed structured output or violates an output specification.
A.4	Plan–Action Disconnect	Executes actions that do not align with, or fail to realize, the stated plan.
A.5	Premature Termination	Stops execution before the task is complete.
A.6	Looping Behavior	Repeats identical or equivalent actions without meaningful progress.
Verification		
V.1	Context Loss	Fails to preserve or use relevant context during verification or later execution.
V.2	Inadequate Verification	Verifies too narrowly, incorrectly, or not at all; accepts conflicting evidence.
Reflection		
RF.1	Progress Misassessment	Incorrectly assesses whether the task is complete or incomplete.
RF.2	Outcome Misinterpretation	Executes correctly but misreads the resulting feedback or environment state.
RF.3	Causal Misattribution	Notices an error but attributes it to the wrong cause.
RF.4	Reflective Hallucination	Reflects on observations or outcomes that never actually occurred.

repository conventions, available dependencies, or environment limits matter. *Injected behavior:* proceed with a plan that overlooks one concrete constraint visible in the context.

- **PL.4 Impossible Action.** Plans actions that are physically, logically, or procedurally impossible given the current state. *Injection point:* a planning step immediately before an action, especially before editing or running commands. *Injected behavior:* plan a next action whose preconditions are not currently satisfied.

C.2 Reasoning Errors

Reasoning errors keep the observations intact but draw the wrong conclusion from them, which makes them detectable only by comparing a step against the evidence available at that point.

- **R.1 Hallucination.** Generates claims not grounded in retrieved observations, tool outputs, or available evidence. *Injection point:* a reasoning step after some evidence has been observed but before the fix is settled. *Injected behavior:* introduce one unsupported factual claim or assumption while keeping the step locally plausible.
- **R.2 Reasoning Error.** Misapplies correct information through entity confusion, temporal mix-ups, faulty inference, or missed edge cases. *Injection point:* a mid-trajectory reasoning step where observations are available and the agent is choosing a direction. *Injected behavior:* use real context from the trajectory but draw a subtly wrong conclusion from it.
- **R.3 Calculation Error.** Arithmetic, counting, unit conversion, measurement, off-by-one, or rounding mistakes. *Injection point:* a step involving line numbers, counts, ranges, versions, offsets, or numeric test interpretation. *Injected behavior:* make a concrete numeric or boundary mistake that plausibly affects the next action.
- **R.4 Task Misunderstanding.** Misinterprets the user's intent or the task scope. *Injection point:* the first few reasoning or exploration steps, before the core solution path is chosen. *Injected behavior:* reframe the task around a nearby but wrong objective or an overly narrow scope.

C.3 Memory Errors

Memory errors corrupt how a later step uses what the trajectory already established, so recognizing them requires relating distant parts of the same trajectory.

- **M.1 Retrieval Failure.** Fails to retrieve relevant information from conversation history, memory, or previous observations. *Injection point:* a later reasoning step that should use earlier observed files, errors, constraints, or decisions. *Injected behavior:* act as if a relevant earlier fact were unavailable, causing redundant or misdirected work.
- **M.2 False Memory.** Incorrectly recalls facts, conversation history, or prior states that never occurred. *Injection point:* a later reasoning or completion step that summarizes what has been found or done. *Injected behavior:* recall one prior observation, command result, file path, or decision incorrectly.
- **M.3 Incomplete Summary.** Compresses stored information too aggressively, omitting details that later affect the work. *Injection point:* a summary or synthesis reasoning step after multiple observations. *Injected behavior:* summarize the state in a way that drops one important condition, edge case, or file detail.

C.4 Action Errors

Action errors corrupt the executed step itself rather than the reasoning around it, and are the category whose consequences the environment reports most directly.

- **A.1 Tool / Parameter Error.** Uses the correct tool with incorrect, malformed, missing, or unreasonable parameters. *Injection point:* a concrete tool call, especially a file edit, search, or test command. *Injected behavior:* keep the same tool family but alter a path, range, command argument, replacement string, or timeout incorrectly.

- **A.2 Hallucinated Tool or Action.** Invokes non-existent tools, APIs, functions, or actions unavailable in the execution environment. *Injection point:* a tool-selection step where the agent is about to inspect, test, or modify the repository. *Injected behavior:* attempt an unavailable command, API, script, or repository helper through a supported wrapper.
- **A.3 Output Format Error.** Produces malformed structured output, invalid syntax, or output violating a specification. *Injection point:* a step writing code, test data, configuration, or final structured output. *Injected behavior:* create a syntactically or structurally malformed output that reads like a small formatting slip.
- **A.4 Plan–Action Disconnect.** Executes actions that do not align with the intended plan or fail to realize it. *Injection point:* a concrete action shortly after reasoning has selected a plan. *Injected behavior:* perform an action in the right general area but inconsistent with the stated plan.
- **A.5 Premature Termination.** Stops execution before completing the task. *Injection point:* middle or late trajectory, after some investigation or edits but before adequate verification. *Injected behavior:* end with a plausible completion claim while required work or verification remains.
- **A.6 Looping Behavior.** Repeatedly performs identical or equivalent actions without meaningful progress. *Injection point:* a read-only, search, or test step where repeating a stale command is plausible. *Injected behavior:* repeat or minimally vary a previous action although the local context calls for a new one.

C.5 Verification Errors

Verification errors corrupt whether a trajectory checks its own work adequately.

- **V.1 Context Loss.** Fails to preserve or use relevant context during verification or later execution. *Injection point:* late verification or follow-up reasoning after tests, searches, or edits. *Injected behavior:* verify the wrong thing because an earlier constraint or edge case has been lost.
- **V.2 Inadequate Verification.** Fails to verify intermediate or final outputs, verifies incorrectly, or accepts conflicting evidence. *Injection point:* late testing, checking, or finishing steps, preferably the earliest point at which the source trajectory later performs a source edit or a stronger check. *Injected behavior:* run a too-narrow check, apply a misleading test filter, or skip a necessary verification with a plausible rationale, creating false confidence in the edited code; when the source trajectory still contains later corrective work, terminate before that work with a premature completion claim.

C.6 Reflection Errors

Reflection errors leave the observations and the checks themselves intact but corrupt the conclusion drawn from them. They are therefore the counterpart of verification errors: the check runs, but its result is read wrongly, which makes these types detectable only by comparing a step against the observation it claims to interpret.

- **RF.1 Progress Misassessment.** Incorrectly assesses whether the task is complete or incomplete. *Injection point:* late reflection after investigation, edits, or partial verification. *Injected behavior:* claim that enough progress has been made, or that more is needed, contrary to the local evidence.
- **RF.2 Outcome Misinterpretation.** Executes an action correctly but misinterprets the resulting feedback or environment state. *Injection point:* a reasoning step immediately after a tool observation, especially an error or test output. *Injected behavior:* read the preceding observation incorrectly while leaving the observation itself unchanged.
- **RF.3 Causal Misattribution.** Identifies that an error occurred but attributes it to the wrong cause. *Injection point:* a reflection step after failed commands, failed tests, errors, or surprising output. *Injected behavior:* acknowledge the error but choose the wrong cause and a correspondingly wrong debugging direction.

- **RF.4 Reflective Hallucination.** Reflects on observations, execution outcomes, or events that never occurred. *Injection point:* a late reflection or final assessment step referencing prior outcomes. *Injected behavior:* cite one plausible but nonexistent observation or result as the basis for the next step.

D Error-Level Attribution

This appendix gives the tables behind Section 5.3. Every number comes from diffing the corpus a run submitted against the pool that run received, over the 13 configurations and 39 runs of Table 2, at the iteration each run submitted.

Table 11: Three-run agreement in intervention relative to correct trajectories. "Higher" and "Lower" count types or categories that remain above or below the correct-trajectory rate in all three runs. Each category averages its constituent types within each run. The range reports the minimum and maximum error-intervention rates across runs.

Researcher	Intervened range (%)	Types higher / lower	Categories higher / lower
GLM-5.3 (Max)	53.5–93.5	0 / 2	0 / 0
Claude Sonnet 5 (High)	16.7–98.7	7 / 0	0 / 0
GPT-6-astra (High)	0.0–53.3	0 / 0	0 / 0
GPT-5.6-sol (Medium)	0.2–97.6	1 / 7	0 / 3
GPT-5.6-luna (High)	0.0–16.5	0 / 17	0 / 4
Kimi K3 (Max)	0.4–100.0	0 / 0	0 / 0
Claude Opus 4.8 (High)	38.7–95.4	3 / 2	1 / 0
Claude Opus 5 (High)	56.3–100.0	0 / 0	0 / 0
GPT-5.6-terra (High)	0.0–0.2	0 / 0	0 / 0
DeepSeek V4 Pro (High)	0.2–48.3	0 / 17	0 / 6
Gemini 3.1 Pro Preview (High)	0.0–0.0	0 / 23	0 / 6
GPT-5.6-sol (High)	0.0–97.6	0 / 0	0 / 0
GPT-5.6-sol (Low)	15.2–79.8	16 / 0	6 / 0

Table 12: What each researcher did to the 460 labelled errors, averaged over three runs. "Intervened" includes drops, rewrites, and loss masking. "Correct" is the corresponding rate on the 1,040 correct trajectories, and "Select." is their difference. "Prec." is the share of intervened trajectories carrying an error label, compared with a 30.7% chance rate. The final five columns partition the 460 errors. "Format" denotes mask filling only and is excluded from "Intervened".

Researcher	Intervened (%)	Correct (%)	Select. (pp)	Prec. (%)	Dropped (%)	Rewritten (%)	Masked (%)	Format (%)	Kept (%)
GLM-5.3 (Max)	74.6	66.2	+8.4	47.6	0.1	74.4	0.0	0.0	25.4
Claude Sonnet 5 (High)	70.4	67.7	+2.6	42.8	2.8	62.2	5.4	0.0	29.6
GPT-6-astra (High)	20.6	21.8	−1.2	38.5	5.4	0.0	15.1	0.0	79.4
GPT-5.6-sol (Medium)	32.7	35.5	−2.8	11.9	10.8	21.9	0.0	0.0	67.3
GPT-5.6-luna (High)	5.5	9.0	−3.5	8.9	5.5	0.0	0.0	0.0	94.5
Kimi K3 (Max)	39.2	35.6	+3.6	34.9	0.3	33.3	5.6	0.0	60.8
Claude Opus 4.8 (High)	62.5	54.9	+7.6	46.8	4.9	40.1	17.5	0.0	37.5
Claude Opus 5 (High)	84.7	70.0	+14.7	43.9	0.5	84.1	0.1	0.0	15.3
GPT-5.6-terra (High)	0.1	2.5	−2.4	2.5	0.1	0.0	0.0	0.0	99.9
DeepSeek V4 Pro (High)	23.0	51.8	−28.8	12.0	1.5	21.4	0.0	42.6	34.4
Gemini 3.1 Pro Preview (High)	0.0	3.5	−3.5	0.0	0.0	0.0	0.0	0.0	100.0
GPT-5.6-sol (High)	60.6	41.3	+19.3	31.6	2.7	25.4	32.5	27.8	11.6
GPT-5.6-sol (Low)	49.5	23.6	+25.9	63.6	46.2	0.0	3.3	0.0	50.5
No researcher (Direct)	0.0	0.0	0.0	–	0.0	0.0	0.0	0.0	100.0

Table 13: Per-type intervention across 13 configurations. "Intervened" includes drops, rewrites, and loss masking; "Strong" includes only drops and rewrites. Δ is the difference from correct trajectories, averaged over 39 runs. "Above" reports how often the intervention rate for an error type exceeds that for correct trajectories, across the 13 configuration means and 39 individual runs.

					Above correct	
Code	**Name**	**Intervened (%)**	**Strong (%)**	**Δ (pp)**	**Config**	**Run**
Planning						
PL.1	Ineffective Planning	41.0	35.0	+3.8	7/13	11/39
PL.2	Goal Drift	45.1	39.1	+7.9	8/13	15/39
PL.3	Constraint Ignorance	41.2	34.9	+4.0	7/13	9/39
PL.4	Impossible Action	36.8	30.6	−0.4	7/13	16/39
Reasoning						
R.1	Hallucination	36.2	29.9	−1.0	6/13	17/39
R.2	Reasoning Error	38.2	31.4	+1.0	8/13	16/39
R.3	Calculation Error	38.1	30.8	+0.9	7/13	15/39
R.4	Task Misunderstanding	45.4	39.2	+8.2	8/13	17/39
Memory						
M.1	Retrieval Failure	38.8	32.3	+1.7	7/13	16/39
M.2	False Memory	44.9	38.1	+7.7	8/13	15/39
M.3	Incomplete Summary	44.6	38.7	+7.4	7/13	17/39
Action						
A.1	Tool / Parameter Error	39.5	33.5	+2.3	6/13	16/39
A.2	Hallucinated Tool	42.8	36.0	+5.6	7/13	13/39
A.3	Output Format Error	43.3	36.4	+6.1	7/13	11/39
A.4	Plan–Action Disconnect	41.7	35.9	+4.5	8/13	17/39
A.5	Premature Termination	30.8	24.9	−6.4	1/13	3/39
A.6	Looping Behavior	46.7	39.4	+9.5	8/13	11/39
Verification						
V.1	Context Loss	35.6	29.9	−1.5	6/13	12/39
V.2	Inadequate Verification	35.1	29.7	−2.1	4/13	15/39
Reflection						
RF.1	Progress Misassessment	31.0	25.5	−6.2	2/13	7/39
RF.2	Outcome Misinterpretation	43.5	39.1	+6.3	7/13	13/39
RF.3	Causal Misattribution	40.5	36.0	+3.3	6/13	12/39
RF.4	Reflective Hallucination	45.0	39.0	+7.8	7/13	16/39

Table 14: Error types ranked by intervention relative to correct trajectories. "Error–Correct" is the within-run difference between each error type and the 1,040 correct trajectories, averaged over 39 runs. Values are rounded to one decimal place.

Rank	Code	Error type	Intervened (%)	Error–Correct (pp)
1	A.6	Looping Behavior	46.7	+9.5
2	R.4	Task Misunderstanding	45.4	+8.2
3	PL.2	Goal Drift	45.1	+7.9
4	RF.4	Reflective Hallucination	45.0	+7.8
5	M.2	False Memory	44.9	+7.7
6	M.3	Incomplete Summary	44.6	+7.4
7	RF.2	Outcome Misinterpretation	43.5	+6.3
8	A.3	Output Format Error	43.3	+6.1
9	A.2	Hallucinated Tool	42.8	+5.6
10	A.4	Plan–Action Disconnect	41.7	+4.5
11	PL.3	Constraint Ignorance	41.2	+4.0
12	PL.1	Ineffective Planning	41.0	+3.8
13	RF.3	Causal Misattribution	40.5	+3.3
14	A.1	Tool / Parameter Error	39.5	+2.3
15	M.1	Retrieval Failure	38.8	+1.7
16	R.2	Reasoning Error	38.2	+1.0
17	R.3	Calculation Error	38.1	+0.9
18	PL.4	Impossible Action	36.8	−0.4
19	R.1	Hallucination	36.2	−1.0
20	V.1	Context Loss	35.6	−1.5
21	V.2	Inadequate Verification	35.1	−2.1
22	RF.1	Progress Misassessment	31.0	−6.2
23	A.5	Premature Termination	30.8	−6.4

Table 15: Error categories ranked by intervention relative to correct trajectories. Each category rate averages its constituent error types, equally weighted because each contains 20 trajectories per run. The within-run difference from the 1,040 correct trajectories is averaged over 39 runs and rounded to one decimal place.

Rank	Error category	Types	Intervened (%)	Error–Correct (pp)
1	Memory	3	42.8	+5.6
2	Planning	4	41.0	+3.8
3	Action	6	40.8	+3.6
4	Reflection	4	40.0	+2.8
5	Reasoning	4	39.5	+2.3
6	Verification	2	35.4	−1.8

E HUMAN RESEARCHER REFERENCE

The human reference was produced by a junior researcher under the same constraints as the agent researchers: the same mixed-quality input pool, 25-hour research budget, provided rollout model, fixed training and evaluation services, and access only to public-split feedback. The human researcher did not have access to the injected error labels or the hidden evaluation split.

The researcher first cleaned the source trajectories by removing clearly erroneous spans while preserving useful surrounding context, and then wrote scripts to repair errors in the trajectory data. To expand the SWE training set, the researcher constructed additional SWE tasks from publicly available data under the same task-source and evaluation-split restrictions, and used the provided rollout model to synthesize 1,000 additional trajectories. The final submitted dataset contained 2,500 trajectories. Training and evaluation used the same fixed benchmark pipeline, yielding the human-reference scores reported in Table 2.

F TRAINING DETAILS

Recipe. Every training request runs LLaMA-Factory (Zheng et al., 2024) full-parameter SFT on Qwen3-8B (Team, 2025b) with DeepSpeed ZeRO-3 (Aminabadi et al., 2022) and bf16 across four A800 80GB GPUs: 2 epochs, learning rate 1×10^{-4} with a cosine schedule and 10% warmup, weight decay 0.01, gradient-norm clipping at 1.0, per-device batch size 1 with gradient accumulation 8, and a 64K-token cutoff with YaRN (Peng et al., 2024) rope scaling. The configuration ships with the harness and is applied verbatim to every request, so the dataset is the only thing that varies between training runs.

Supervision format and masking. Datasets are staged as ShareGPT rows, one per trajectory. Per-turn loss masking is honored through the `turn_mask` field of each message, so a researcher can keep a failed assistant turn in context while excluding it from the supervised target. Masks on non-assistant messages do not change the supervised target.

G EVALUATION SUITE AND SEED CORPUS

The complete evaluation protocol used in the main experiments is listed in Table 16. Every researcher configuration is scored with the same splits, runners, verifiers, and decoding settings, and no configuration can alter them: evaluation is a fixed service of the harness (Section 3.3). The RSI-WILD splits measure the targeted capability of each track, and the four general benchmarks are held out entirely from the research loop and evaluated once, on the submitted checkpoint, to expose data strategies that buy target-task gains by damaging the base model.

SWE splits. Both splits are drawn from the 500 instances of SWE-bench Verified (Jimenez et al., 2024). Instances are ordered by the SHA-256 hash of their instance id under a fixed seed string; the first 100 form the public split the researcher sees after every training run, the next 100 form the hidden split used for the final score, and the remaining 300 are unused. The manifest is frozen and shipped with the benchmark, so all configurations are compared on identical instances. Checkpoints are served with vLLM (Kwon et al., 2023) and rolled out by OpenHands (Wang et al., 2025), CodeActAgent (Wang et al., 2024) at temperature 0 with at most 100 steps per instance, and the resulting patches are scored by the official SWE-bench harness (Jimenez et al., 2024).

Terminal splits. The terminal track partitions the 100 tasks of TBLite (OpenThoughts-Agent team, 2026) into two disjoint 50-task splits using a fixed seeded ordering. The first 50 tasks form the public split used for research feedback, while the remaining 50 form the hidden split used only for the final score. We use this partition rather than Terminal-Bench 2.0 because the latter is too difficult for the Qwen3-8B model, yielding little useful resolution for comparing data-research strategies. Both splits are evaluated with the Terminus-2 agent against the checkpoint served by vLLM, and each task is scored by its own verifier.

Seed corpus. Each track draws its trajectories from one corpus of harness-verified successes. The SWE corpus is 5,009 publicly released SWE-Lego (Tao et al., 2026) rollouts in SWE-bench format;

the terminal corpus is 5,795 TMAX (Ivison et al., 2026) rollouts over 1,136 distinct terminal tasks, kept only where the task was solved. Pool instances are sampled from these corpora under seeded hashing and are disjoint from the evaluation tasks above.

Table 16: Evaluation suite. The same splits, runners, verifiers, and decoding settings are used for every researcher configuration. AIME uses four decoding runs per problem and reports their mean. The two SWE splits are disjoint subsets of SWE-bench Verified, the two terminal splits are disjoint subsets of TBLite, and all four general benchmarks are run in full.

Benchmark	Items	Coverage	Runner	Metric	Decoding
RSI-WILD *SWE track*					
SWE-bench Verified (public)	100	20.0% of 500	CodeActAgent, $\leq$100 steps	Resolved rate	Greedy, 64K context
SWE-bench Verified (hidden)	100	20.0% of 500	CodeActAgent, $\leq$100 steps	Resolved rate	Greedy, 64K context
RSI-WILD *terminal track*					
TBLite (public)	50	50.0% of 100	Terminus-2	Task success rate	Terminus-2 defaults
TBLite (hidden)	50	50.0% of 100	Terminus-2	Task success rate	Terminus-2 defaults
General benchmarks (submitted checkpoint only)					
HumanEval	164	Full set	Sandboxed execution	pass@1	Greedy, 4K output
GPQA Diamond	198	Full set	Answer matching, choices shuffled	Accuracy	Greedy, 16K, thinking on
AIME 2026	30 (4×)	Full set	Answer matching	Mean accuracy over 4 runs	T=0.6, top-p 0.95, top-k 20
IFBench	300	Full set	IFBench verifiers	Prompt-level loose accuracy	Greedy, 8K, thinking on

H RESEARCH AGENT PROMPT

The following prompt is given to every research agent in the main experiments.

System Prompt for the Research Agent

Role

You are a long-lived autonomous SFT data researcher. Your goal is to build high-quality training data, start experiments, inspect public evaluation results, and submit the model most likely to generalize.

Hard Control Rule

Training, evaluation, and agent work share a single 25-hour budget. Training begins only when you explicitly run `/workspace/agentctl train`, and every successful training run automatically triggers the full public evaluation. Training parameters are fixed, so focus entirely on data strategy. The final evaluation uses a separate dataset, so optimizing for or exploiting the public evaluation will not help. Training opportunities are limited and valuable. Thoroughly refine and validate the data before starting a run. Submit the model version you consider most reliable once the evidence is sufficient or the remaining time budget is running low.

Suggested Data Research Strategies

These are non-exhaustive suggestions. Select or combine them according to the data and the hypothesis being tested.

- **Trajectory cleaning:** Remove low-quality, malformed, repetitive, or duplicated trajectories.

- **Trajectory repair:** Correct useful trajectories with localized defects or incomplete endings.
- **Task synthesis and trajectory rollout:** Create new tasks for meaningful coverage gaps, then use the small model to generate candidate rollout trajectories.

Small Model helper
The small model can be used to generate candidate trajectory rollouts or continuations. Call it with:

```
/workspace/rollout \
--system "Generate one grounded candidate
software-engineering trajectory continuation."
\
--prompt "Continue this source trajectory from its
last grounded observation:  ..."
```

Commands
Show live controller status:

```
/workspace/agentctl status
```

Reports the current action, remaining budget, experiment states, latest result, and submission state.
List experiments:

```
/workspace/agentctl experiments
```

Shows each experiment's training status, evaluation status, and evaluation count.
Train:

```
/workspace/agentctl train
```

Trains with `/workspace/outputs/train.json` and automatically evaluates all 100 public validation questions.
Submit:

```
/workspace/agentctl submit \
--experiment 0 \
--notes "why this model should generalize best"
```

Submits the selected experiment and ends the session after success.

Record
Record the research process in /workspace/outputs/researcher report.md.

Completion
Submit the model version you consider best with the submission command.

Figure 12: The system prompt given to research agents.