WHAT IS MISSING FROM AI POST-TRAINING AI: AN EMPIRICAL ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents can now post-train an LLM end-to-end, raising the prospect of recursive self-improvement (RSI). Yet this progress is measured by aggregate benchmark scores, which cannot tell whether an agent executes a fixed plan well or strategically revises the plan when it fails. We separate these two capabilities: *execution-level* capability, iterating within an established training strategy, and *strategy-level* capability, revising that strategy as experimental evidence accumulates. Analyzing 1,338 post-training trajectories of frontier agents, we find that **agents reliably execute post-training but lock into a default strategy**, which follows the agent rather than the task, and only 2.1% of transitions between adjacent training runs ever change strategy. We then test whether the agent lacks experience, reasoning, or the decision to switch. (1) Experience improves execution but not the strategy. (2) Additional reasoning compute yields front-loaded gains on easier tasks but refines, rather than revises, the committed strategy. (3) Human review before training changes *which* strategy the agent locks into, not *whether* it locks in, whereas a single mid-run instruction outperforms the agent's own continuation by up to 17.44 points under the same budget. In conclusion, what the agent lacks is **the decision to reopen a committed strategy and try another one**. Realizing RSI therefore calls for interaction protocols and training signals that make strategy revision an explicit, rewarded decision. [1]

1 INTRODUCTION

Recent advances in large language model (LLM) agents have rapidly expanded their role in AI research and development (Karpathy, 2026). Frontier agents can now optimize kernels, train models, and manage complex engineering pipelines over hours-long horizons (Krishnan, 2025; Gonzalez et al., 2026; Sun et al., 2026; Patwardhan et al., 2025). This progress raises the prospect of recursive self-improvement (RSI), in which AI systems improve AI systems themselves (Lu et al., 2026a). LLM post-training has become a concrete testbed for this prospect, as it is supported by mature infrastructure and standardized benchmarks. PostTrainBench (Rank et al., 2026) formalizes this setting and shows that frontier agents can already post-train an LLM end to end, and a growing body of work hands individual stages of post-training to agents, including data preparation (Du et al., 2026b; Deng et al., 2026), experiment configuration (Guo et al., 2026b), and training-strategy search (Fang et al., 2026; Chen et al., 2026b), each reporting gains over hand-designed counterparts.

These gains, however, are reported as aggregate benchmark scores, which cannot tell *why* a run succeeds or stalls. A run may improve because the agent executes a fixed plan well, or because it strategically changes the plan when the plan stops working. The distinction matters for RSI: the former improves a pipeline, whereas only the latter improves how the agent conducts research. We therefore separate two levels of capability that current discussions of RSI tend to conflate: (1) *execution-level* capability, iterating within an established training strategy, such as fixing bugs, tuning hyperparameters, and reformatting data; and (2) *strategy-level* capability, revising the high-level choice of training algorithm, data source, or training stages as experimental evidence accumulates.

We first measure both levels at scale. Analyzing 1,338 PostTrainBench trajectories spanning seven benchmarks, four base models, and three agent frameworks (Rank et al., 2026), we find that **agents**

[1] We release our code at `https://anonymous.4open.science/r/AI-PostTraining-AI`.

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.

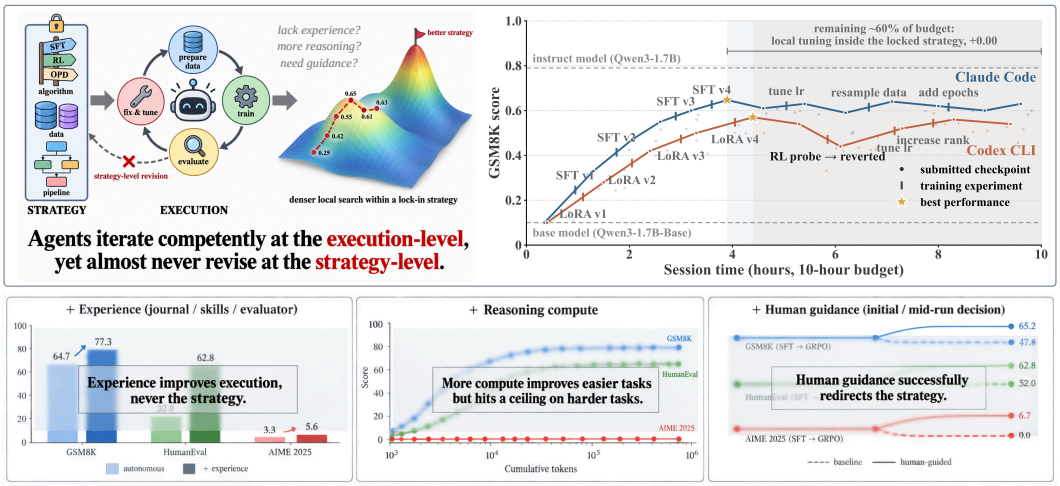


Figure 1: **Overview. Top:** agents iterate reliably at the execution level yet almost never revise the committed strategy (Section 3). **Bottom:** experience and reasoning compute improve execution but leave the strategy unchanged, whereas human guidance redirects it (Section 4).

reliably execute post-training: they complete the full pipeline, diagnose and repair failures, and improve over the base model on every benchmark. Yet **each agent locks into a default strategy** before any experimental evidence arrives, and the default follows the agent rather than the task. For example, on the same benchmarks, Claude Code mostly starts with full-parameter supervised fine-tuning (SFT), whereas Codex CLI mostly starts with parameter-efficient fine-tuning (PEFT). Once training begins, only 74 of 3,557 adjacent training pairs (2.1%) ever change strategy, and the remaining budget goes into local adjustments within the initial choice (Figure 1). As a result, the execution-level gap shrinks steadily, while the strategy-level gap is fixed at the first experiment.

We then ask what unlocks the strategy-level capability. Revising a strategy often requires the *experience* to tell when it is failing and what to try next, the *reasoning* to weigh an alternative against it, and the *decision* to actually switch (Chi et al., 2026; Liu et al., 2026; Guo et al., 2026a). We test each in turn on GSM8K, HumanEval, and AIME 2025, three benchmarks of increasing difficulty: (1) *Missing experience?* We equip the agent with an experiment journal, a skill library distilled from open-source post-training frameworks, and an evaluator agent that diagnoses checkpoints on request. The result shows that execution improves substantially, closing 75.1% of the gap to the official instruct model, yet the strategy does not move: the agent adopts all 22 execution-level suggestions from the evaluator but none of the 21 strategy-level ones. (2) *Insufficient reasoning?* Tracking the score against cumulative token usage, we find that gains on easier tasks are front-loaded; on AIME 2025, even in the best run, roughly 14M additional tokens buy only one extra problem, within evaluation variance. The extra compute goes into refining the committed strategy, not revising it. (3) *Missing decision?* We let a human make this decision at two points. A review before training changes *which* strategy the agent locks into, but not *whether* it locks in. A single mid-run instruction to switch, forked from the same checkpoint under the same remaining budget, outperforms the agent's own continuation on all three benchmarks, by up to 17.44 points.

Our experiments locate what is missing. It is not the ability to carry out a different strategy: under guidance, the agent implements an unfamiliar strategy and even extends it on its own. Nor is it a lack of resource: experience and reasoning compute both improve execution, yet neither leads the agent to leave its committed strategy. What is missing from AI post-training AI is **the decision to reopen a committed strategy and try another one**. Human guidance works precisely because it makes this decision on the agent's behalf; nothing we supplied leads the agent to make it on its own.

In conclusion, the prevailing picture of RSI assumes a loop that closes globally: propose a strategy, run experiments, interpret the results, revise the strategy, and repeat. Our study shows that current agents close this loop only at the execution level and leave it open at the strategy level (Figure 7). Therefore, realizing RSI calls for interaction protocols and training signals that make strategy revision an explicit, rewarded decision rather than an implicit option buried in a long context.

2 PRELIMINARIES

2.1 A TWO-LEVEL CAPABILITY FRAMEWORK

To separate the two levels of capability, we decompose every training experiment into a *strategy* and its *execution* configuration. Formally, we model the strategy space as $\mathcal{S} := \mathcal{P} \times \mathcal{D} \times \mathcal{G}$, where $\mathcal{P}$ ranges over canonical training algorithms (e.g., SFT, and reinforcement learning (RL)), $\mathcal{D}$ represents the data source (e.g., curated, self-generated, or mixed), and $\mathcal{G}$ describes the training stages (e.g., a single-stage SFT vs. SFT followed by RL). The full taxonomy is provided in Appendix A. Let $\mathcal{X}$ denote the space of execution-level configurations, including data formatting, hyperparameters, and implementation details. An agent trajectory is denoted as a sequence $\tau = \{(s_t, x_t, r_t)\}_{t=1}^{T}$.

The t-th experiment runs strategy $s_t \in \mathcal{S}$ under configuration $x_t \in \mathcal{X}$ and yields an evaluation result r_t. A transition $t \to t{+}1$ is a *strategy change* iff $s_{t+1} \neq s_t$, i.e., it alters the training algorithm, the data source, or the training stages, and an *execution change* otherwise ($s_{t+1} = s_t$, $x_{t+1} \neq x_t$).

• **Execution-Level Capability:** Execution-level actions move within $\mathcal{X}$ while holding s_t fixed, including formatting data, tuning hyperparameters, and debugging the implementation. This capability determines how reliably an agent turns a selected strategy into a working pipeline.

• **Strategy-Level Capability:** Strategy-level actions move within $\mathcal{S}$ by switching the training algorithm, changing the data source, or modifying the training stages. This capability determines whether the agent revises its high-level judgment ($s_{t+1} \neq s_t$) in light of accumulated evidence $r_{1:t}$.

The Strategy–Execution Gap. Let $V(s, x)$ denote the benchmark score of a strategy s with configuration x, and write $V^\star(s) := \max_{x \in \mathcal{X}} V(s, x)$ and $V^\star := \max_{s \in \mathcal{S}} V^\star(s)$. For a trajectory locked into its initial strategy ($\forall t, s_t = s_1$) with best score $V(\tau) := \max_t r_t$, the total gap decomposes as

$$\underbrace{V^\star - V(\tau)}_{\text{total gap}} = \underbrace{V^\star - V^\star(s_1)}_{\text{strategy-level gap}} + \underbrace{V^\star(s_1) - V(\tau)}_{\text{execution-level gap}}. \tag{1}$$

In the following sections, we find that agents make robust execution-level progress that steadily reduces the *execution-level gap* (Finding 1) but freeze the *strategy-level gap* at $t = 1$ (Finding 2).

2.2 DIAGNOSING POST-TRAINING THROUGH AGENT TRAJECTORIES

We analyze 1,338 agent trajectories from PostTrainBench (Rank et al., 2026), covering seven benchmarks, four base models, and three agent frameworks (Claude Code, Codex CLI, and OpenCode), each produced under a 10-hour budget on a single H100 GPU. For each trajectory $\tau = \{(s_t, x_t, r_t)\}_{t=1}^{T}$, we count a *training experiment* only when an executed command actually starts a model parameter update, which yields 5,111 verified experiments across the 900 trajectories that launch training. We label each experiment with its strategy state $s_t \in \mathcal{S}$ and classify each pair of adjacent experiments as either a strategy change ($s_{t+1} \neq s_t$) or an execution-level adjustment ($s_{t+1} = s_t$, $x_{t+1} \neq x_t$). Labels are produced by rule-based scripts together with an LLM and subsequently reviewed by the authors. The full annotation protocol is detailed in Appendix A.

3 EFFECTIVENESS AND LIMITATIONS OF AI POST-TRAINING AI

This section provides the trajectory-level evidence for the two-level capability framework and the strategy–execution gap of equation 1, quantified over the released agent trajectories.

3.1 FINDING 1: AGENTS ARE RELIABLE EXECUTORS

Finding 1 (Reliable execution) *Frontier agents reliably instantiate their chosen strategy: they complete the full training pipeline and improve over the base model on every benchmark.*

Figure 2 shows the score distribution of all submitted checkpoints. Across 1,338 trajectories, agents sustain an average of 3.82 training runs and 13.80 evaluations per trajectory. They complete the full post-training pipeline, from data preparation through training and evaluation to checkpoint submission, and improve over the base model. They also perform technically meaningful diagnosis and

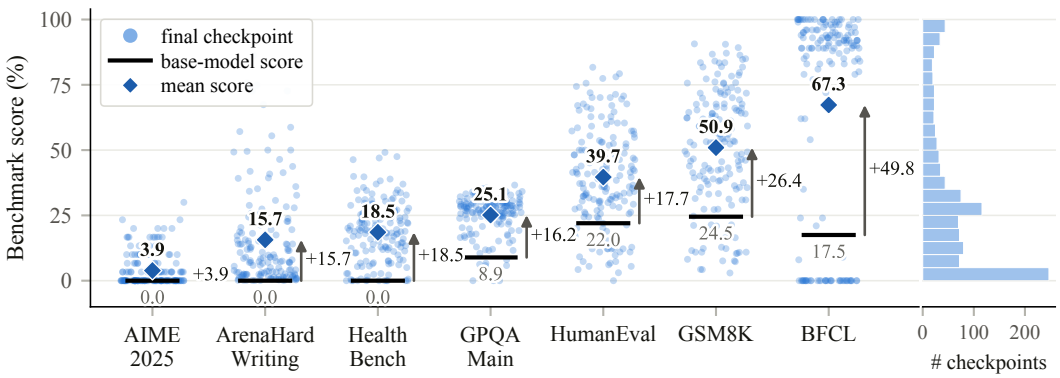


Figure 2: Benchmark scores of the submitted checkpoints averaged over four base models: agents reliably execute post-training pipelines and improve over the base model across seven benchmarks.

repair that move within $\mathcal{X}$. For example, in the GSM8K run shown in Figure 1, Claude Code trains four successive SFT versions that effectively lift the score from the 0.10 base level to 0.65 within the first four hours, and keeps iterating afterwards by tuning the learning rate, resampling data, and adding training epochs. In short, **frontier agents are reliable executors of post-training**. The execution-level gap in equation 1 leaves limited headroom; the main constraint lies elsewhere.

3.2 Finding 2: Agents Lock into Default Strategies

For a trajectory with $T \geq 2$ training experiments, we define the *switch rate* $\rho(\tau)$ as the fraction of adjacent experiment pairs that change strategy. For an agent a, the *default-strategy concentration* κ_a is the mass its initial-strategy distribution $\pi_a(s) := \Pr[\, s_1 = s \mid a \,]$ places on its modal strategy:

Table 1: Default-strategy concentration κ_a and switch rate ρ_a per agent (Appendix A).

Agent (#Traj.)	Default Strategy (κ_a)		Switch Rate (ρ_a)
Claude Code (575)	Full SFT	71.9%	54/1,203 (4.5%)
Codex CLI (369)	PEFT	89.5%	15/943 (1.6%)
OpenCode (394)	Full SFT	66.4%	5/1,411 (0.4%)
Overall (1,338)		76.7%	74/3,557 (2.1%)

$$\rho(\tau) := \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbf{1}[\, s_{t+1} \neq s_t \,], \qquad \kappa_a := \max_{s \in \mathcal{S}} \pi_a(s). \tag{2}$$

We write $\bar{\rho}$ for the pooled switch rate over all adjacent training pairs, and ρ_a for the same pooled switch rate restricted to the verified pairs of agent a's trajectories. A trajectory with $\rho(\tau) = 0$ is completely *locked in*, where its strategy-level gap in equation 1 is fixed at $t = 1$.

> **Finding 2 (Strategy lock-in)** *Agents commit to a default strategy (one that follows the agent rather than the task), and spend the remaining budget on local adjustments within it.*

The default strategy follows the agent, not the task. Agents typically lock into a default training strategy at the beginning of a run[2], and different agents lock into different defaults on the same task (Table 1). Claude Code concentrates on full-parameter SFT, while Codex CLI concentrates on PEFT. The divergence is systematic: in every combination of the seven benchmarks and four base models, Claude Code shows a higher full-SFT share and Codex CLI a higher PEFT share (Table 4).

Once training begins, the strategy rarely changes. Among the 3,557 adjacent training pairs, only 74 (2.1%) ever probe an alternative (decomposed by dimension in Table 5). The rate stays low even where the default strategy nearly fails. For example, on AIME 2025, final scores remain close to zero, yet the highest per-agent switch rate is only 10.6%. Switch rates do vary mildly with the task (Claude Code reaches about 11% on AIME 2025 but stays at 0% on BFCL), but no benchmark elicits systematic search over strategies. The remaining budget goes into denser local search within $\mathcal{X}$, such as tuning learning rates and changing chat templates, rather than search over $\mathcal{S}$.

[2]The released trajectories do not expose a uniform wall-clock timestamp. "The beginning of a run" denotes the pre-update planning phase, not a pooled estimate of time-to-lock.

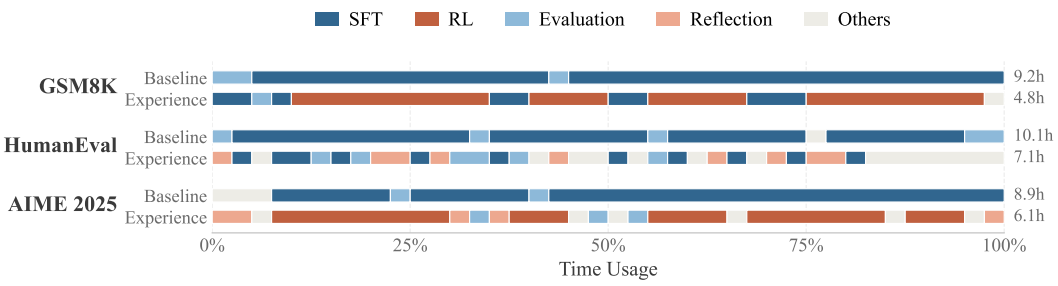


Figure 3: Time usage of representative runs. The autonomous baseline spends most of the budget in SFT, while the experience-driven agent alternates training with evaluation and reflection.

4 What is Missing from AI Post-Training AI

The previous section shows that agents execute post-training reliably yet lock into a default strategy and iterate within it. This section asks **what unlocks the strategy-level capability**. Revising a committed strategy requires three things: the *experience* to recognize that the current strategy is failing and to know what to try next, the *reasoning* to weigh the alternative against the committed choice, and the *decision* to act on it. Strategy lock-in may stem from a shortfall at any stage, so we examine them in turn: whether the agent lacks experience (Section 4.1), whether its reasoning compute is insufficient (Section 4.2), and whether the decision itself is missing (Section 4.3).

Experimental Setup. We adopt the standard PostTrainBench setting with Qwen3-1.7B-Base (Yang et al., 2025a) as the base model. We evaluate on three benchmarks of increasing difficulty, namely GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and AIME 2025. The autonomous baselines include three frontier agents: Claude Code powered by Opus 4.6 or GLM-5.2, and Codex CLI powered by GPT-5.2. All interventions build on Claude Code with Opus 4.6. [3] Each configuration is run three times independently under a 10-hour budget on four NVIDIA A800 GPUs; the system prompt, base model, hardware, and evaluation protocol remain fixed within each comparison.

4.1 Is Experience Missing?

Prior work attributes the bottleneck of AI post-training AI to the lack of practical experience for experimental decisions (Abbasi, 2026). We test this explanation with an experience-driven framework built around three components (Appendix C): an **experiment journal** that persists plans, evaluation results, observations, and lessons across iterations, keeping earlier evidence accessible as the context grows; a **skill library** that distills documentation, training recipes, and known issues and pitfalls from widely used training frameworks into reusable skills; and an **evaluator agent** that analyzes each requested checkpoint and returns compact diagnoses with actionable suggestions.

4.1.1 Experience Improves Execution-Level Capability

Table 2 reports the benchmark scores averaged over three independent runs. **The experience-driven framework outperforms the autonomous baselines on all three benchmarks**, closing 75.1% of the gap to the official instruct model versus 48.6% for the same agent running autonomously. The agent actively uses the provided resources: it draws on skills 18–60 times per run, and records plans, evaluation results, and lessons in the experiment journal (Appendix C.2). As shown in Figure 3, the experience-driven agent interleaves shorter training runs with dense evaluation and reflection, completing more experiments within the same budget and converting each into recorded evidence for the next. In summary, the supplied experience **systematically improves execution** across benchmarks. With experience, the agent executes experiments and diagnoses failures more efficiently.

[3] In preliminary runs on Qwen3-1.7B-Base and the three benchmarks, Opus 4.6 is reliable than Opus 4.8, Opus 5, and Fable 5: the newer models either exhibit high variance across runs or introduce data contamination (e.g., train-on-test), which falls outside the scope of this study (Appendix E.3).

Table 2: Benchmark scores (%) under the controlled setting with Qwen3-1.7B-Base as the base model, reported as mean ± one standard deviation over three independent runs.

	Benchmark			Aggregate	
Setting	**GSM8K**	**HumanEval**	**AIME 2025**	**Avg.**	**Gap Closed**
Base model	10.84	5.48	0.00	5.44	0.0%
Official instruct model	88.70	66.46	33.33	62.83	100.0%
Opus 4.6 (Claude Code)	64.70 ±9.6 (+53.9)	32.00 ±10.4 (+26.5)	3.33 ±0.0 (+3.3)	33.34	48.6%
GLM-5.2 (Claude Code)	49.51 ±7.2 (+38.7)	44.51 ±9.8 (+39.0)	3.33 ±0.0 (+3.3)	32.45	47.1%
GPT-5.2 (Codex CLI)	43.44 ±4.1 (+32.6)	13.41 ±8.7 (+7.9)	0.00 ±0.0 (0.0)	18.95	23.5%
Experience-driven (Opus 4.6)	**77.30** ±3.8 (+66.5)	**62.80** ±6.1 (+57.3)	**5.56** ±1.6 (+5.6)	**48.55**	75.1%
w/o experiment journal	74.50 ±4.5 (−2.8)	50.20 ±7.4 (−12.6)	4.44 ±1.6 (−1.1)	43.05	65.5%
w/o skill library	73.10 ±5.2 (−4.2)	54.50 ±6.8 (−8.3)	3.33 ±0.0 (−2.2)	43.64	66.6%
w/o evaluator agent	68.20 ±6.0 (−9.1)	42.60 ±8.2 (−20.2)	3.33 ±0.0 (−2.2)	38.04	56.8%

4.1.2 Strategy-Level Capability Remains Limited

Despite the execution-level progress, **strategy-level revisions remain rare**. The agent still locks into its default strategy, producing 14 consecutive SFT variants on HumanEval and tuning hyperparameters back and forth on AIME 2025 (Figure 8). The experience-driven framework consistently exposes critical evidence about the current state, but interestingly, the agent acts only on its execution-level part. On HumanEval, the evaluator agent repeatedly suggests an RL stage, and the journal records that SFT has plateaued; the main agent nevertheless writes an RL training script but never launches it. We illustrate this pattern in Figure 4. The main agent adopts all 22 execution-level suggestions, concerning reward shaping, hyperparameter tuning, and output formatting, but none of the 21 strategy-level suggestions, such as switching to RL or adding an SFT stage.

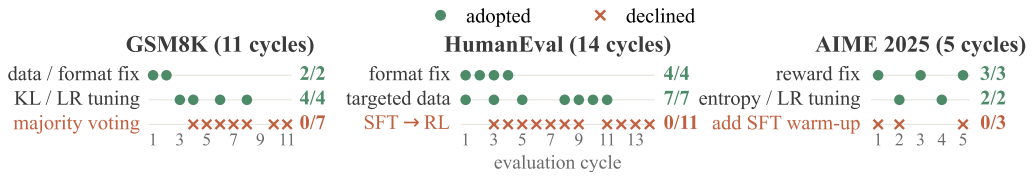


Figure 4: Across evaluation cycles, the main agent implements every execution-level suggestion (22/22) but declines every suggestion that departs from its current strategy (0/21).

In summary, the experience-driven framework consistently preserves and clarifies the experimental evidence, and reliably improves how the agent executes and repairs the current training strategy. However, **the agent's responses to this evidence remain at the execution level**. Negative evidence triggers further adjustments within the strategy, rather than a revision of the strategy itself, especially after the agent has invested substantial time and compute in the current strategy.

4.2 Is Reasoning Compute Insufficient?

A common expectation is that scaling reasoning compute enables the agent to search beyond its default and reach a better strategy (Wang et al., 2026; Zhu et al., 2025a). To examine the marginal value of this compute, we systematically track the agent's cumulative token usage throughout each run. Figure 5 presents how the benchmark score grows with the agent's token consumption.

More compute improves easier tasks. The experience-driven agent consumes roughly 1.8–9.2× more tokens than the autonomous baseline. On GSM8K and HumanEval, the additional tokens convert into notable performance gains (3.7M and 0.9M tokens per point of improvement on average). However, these gains are front-loaded. On GSM8K, roughly 90% of improvement is realized within the first half of its token budget, and the remaining ∼11M tokens buy only +2.1 points (on HumanEval, +51.8 points are in the first half with only +9.8 points afterwards).

Compute scaling hits a ceiling on harder tasks. On AIME 2025, the hardest benchmark in our setting, roughly 14M additional tokens buy only a single extra problem, which lies within evaluation variance. The extra compute is spent exactly where Finding 2 predicts: on denser local search within the committed strategy. Scaling agent context and reasoning compute therefore has a clear ceiling: once the task demands a strategy the agent did not start with, additional tokens buy more refinement, not better decisions. We present this finding as a practical takeaway for agent system developers: additional compute can improve execution, but it quickly hits the ceiling on more challenging tasks.

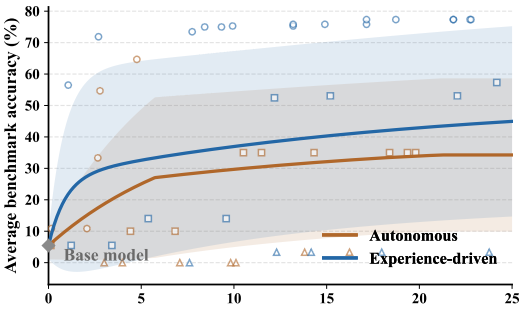


Figure 5: Average benchmark score against the main agent's cumulative token consumption (○ GSM8K, □ HumanEval, △ AIME 2025)

4.3 Is the Decision Missing?

In previous sections, experience provides the agent with evidence about the current strategy and knowledge of potential alternatives, and additional reasoning compute pushes the current strategy toward its ceiling, yet the agent still stays in it. What remains is the decision to reopen the committed strategy and try another one. We therefore include human guidance at two decision points: a human review at the initial choice of strategy, and a single instruction to switch strategy mid-run.

4.3.1 Human Guidance at the Initial Decision

At the initial decision point before training begins, a human reviewer iteratively approves the agent's proposed strategy, or requests a revision with an explicit rationale. Once the strategy is approved, the runs proceed fully autonomously under the experience-driven framework of Section 4.1.

The review redirects the initial strategy, and **the agent then locks into the new strategy just as it locks into its own default**. On AIME 2025, the agent first proposes an SFT pipeline, and the reviewer asks that the main budget go to RL. Once execution begins, the agent inspects the base model, finds the required output format already attainable without SFT, and skips the warm-up altogether. This indicates that the agent understands, implements, and even extends on its own a strategy different from its own proposal. The improvement, however, is not sustained. The runs reach their best score early and then iterate within the new strategy without further gains (Figure 6a). The agent tunes hyperparameters back and forth, repeats attempts that the journal already records as unsuccessful, and devotes most later training to the local question of how to resume from an earlier checkpoint (Appendix D.1.2). This pattern is consistent with a central challenge of long-horizon agentic tasks (Barke et al., 2026; Lin et al., 2026): as the context grows, recent observations and system logs come to dominate the agent's deliberation, crowding out the strategy-level question of whether the current strategy still deserves the remaining budget. Therefore, human guidance at the initial decision actually changes *which* strategy the agent locks into, but not *whether* it locks in.

4.3.2 Human Guidance at a Mid-Run Decision

Since guidance at the initial decision does not keep the agent from locking in, we next test whether a single instruction issued mid-run can reopen a strategy the agent has already committed to. For each benchmark, we select a mid-run checkpoint from a completed autonomous trajectory, fork the run, and instruct the agent to switch to an alternative strategy. We then compare this guided branch against the agent's own recorded continuation under the same remaining budget (Appendix D.2).

A single mid-run instruction reopens the committed strategy. As shown in Figure 6b, the guided branch reaches a higher final score on all three benchmarks, by up to 17.44 points on GSM8K. A plausible reason is that the shared prefix has already secured the prerequisites of the alternative strategy, such as format compliance from the early SFT, so the switch is well primed at the branch point, whereas the agent's own continuation keeps refining a strategy whose returns have flattened. This result shows that a better route was reachable from the same checkpoint, not that the alternative is optimal or that switching is always beneficial. To rule out the possibility that the agent merely

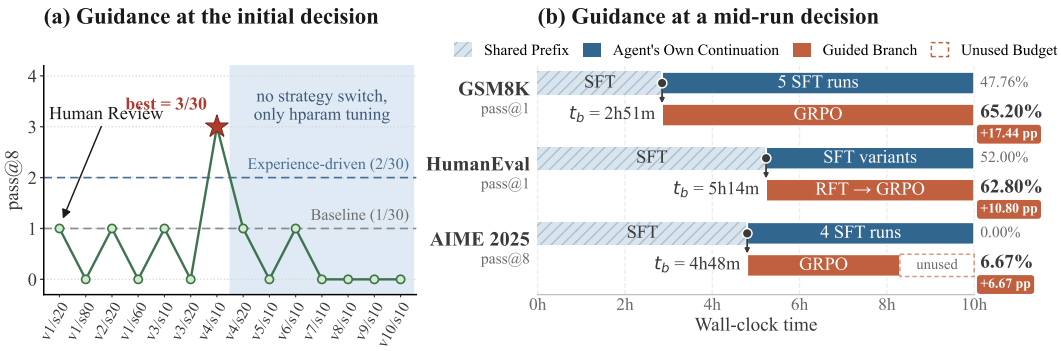


Figure 6: (a) Guidance at the initial decision: the AIME 2025 run peaks early; later iterations (shaded) never recover the peak. (b) Guidance at a mid-run decision: a single instruction to switch strategy outperforms the agent's own continuation under the same remaining budget.

lacks a prompt to reconsider, we then fork each checkpoint with an instruction that asks the agent to reconsider its current strategy without naming an alternative. However, in every such fork, **the agent reaffirms its current strategy** and proceeds as its recorded continuation does (Appendix D.2). Therefore, what the agent lacks is not only the decision point, but also the decision itself.

4.4 Summary

What is missing from AI post-training AI is not the ability to carry out a different strategy. Under human guidance, the agent implements an unfamiliar strategy and even extends it on its own (Section 4.3.1). Nor is it a lack of resources. The experience and reasoning compute both improve execution and close much of the execution-level gap, yet neither leads the agent to leave its committed strategy (Sections 4.1 and 4.2). What is missing is **the decision to reopen a committed strategy and try another one**: human guidance works precisely because it makes this decision on the agent's behalf, and nothing we supplied leads the agent to make it on its own (Section 4.3.2).

5 What This Implies for RSI

In the preceding sections, we find that the RSI loop closes only at the execution level and stays open at the strategy level (Figure 7). We discuss the implications of this finding for progress toward RSI.

Scaling Deepens the Loop That Already Closes. The quality of the initial strategy, rather than the number of iterations or the amount of compute, sets the upper bound of a run. Our compute analysis makes the same point: additional tokens buy denser refinement within the committed strategy but not a better one, so an agent can iterate efficiently yet remain in the wrong strategy. Scaling execution-level autonomy therefore amplifies the loop that already closes and leaves the open one untouched.

Measuring RSI Progress Requires Strategy-Level Metrics. Current RSI benchmarks often report only a final score (Meng et al., 2026; Rank et al., 2026; Lu et al., 2026b; Chi et al., 2026). Under this measure, two agents with the same score may differ in exactly the capability RSI depends on: one chooses the right strategy for the task, whereas the other holds a default that happens to suit it and may fail once the task changes. We therefore suggest that RSI benchmarks report strategy-level metrics alongside the final score. Our analysis already provides three such metrics: the *switch rate* in equation 2, the *adoption rate* of strategy-level suggestions (Figure 4), and the *strategy-level gap* estimated by forking a switched branch from a mid-run checkpoint (Section 4.3.2).

Building RSI Agents That Revise Their Strategy. In the near term, our results favor human guidance at mid-run decision points over a review before training (Section 4.3). In the long term, the decision to revise must come from the agent itself, which calls for two ingredients: (1) an *interaction protocol* that makes "continue or switch" an explicit decision node after each evaluation and before any further action, rather than an implicit option buried in a long context; (2) a *training signal* derived from forked runs (Section 4.3.2), where the "continue" and "switch" branches from the same checkpoint under the same remaining budget naturally form a controlled comparison of

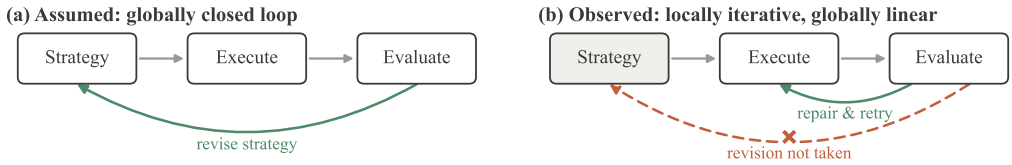


Figure 7: (a) RSI assumes a loop that closes globally in which the evaluation feeds back into the strategy. (b) The observed loops, however, only close at the execution level (repair & retry).

the two actions at that node. Collecting such paired trajectories under this protocol would therefore provide a clean training signal that could unlock the strategy-level capability.

6 Related Work

AI for AI. A growing line of work applies LLM agents to AI research and development (AI R&D) (Karpathy, 2026; Xu et al., 2026b). End-to-end systems carry out the full machine learning research cycle (Yamada et al., 2025; Schmidgall et al., 2025; Tang et al., 2026; Li et al., 2026c). Other agents take on narrower tasks, such as ML engineering (Jiang et al., 2025; Yang et al., 2025b), algorithm discovery (Du et al., 2026a; Chen et al., 2026a), and individual stages of LLM post-training (Kulikov et al., 2026; Du et al., 2026b; Deng et al., 2026; Fang et al., 2026). Benchmarks evaluate such agents on AI R&D (Huang et al., 2024; Chan et al., 2025; Starace et al., 2025) and, more recently, on LLM post-training (Rank et al., 2026; Chen et al., 2026b; Li et al., 2026a). These systems and benchmarks measure or scale how agents *execute* AI development. Rather than adding another component, we ask whether agents also revise the *strategy* that organizes all this search.

Recursive Self-Improvement (RSI). RSI envisions a loop of proposing a strategy, running experiments, interpreting the results, and revising the strategy (Lu et al., 2026a). Self-improving agents pursue this vision by rewriting their own code (Zhang et al., 2026b) or harnesses (Zhang et al., 2026a). Recent benchmarks measure progress toward RSI in data-centric research (Meng et al., 2026), algorithm design (Chi et al., 2026), and agent development (Lu et al., 2026b). Meanwhile, critiques of research agents identify weak scientific taste, implementation drift, memory degradation, missing tacit knowledge, and shallow convergence (Bisht et al., 2026; Trehan & Chopra, 2026; Abbasi, 2026). These benchmarks score the outcome of the loop, and these critiques catalog the shortcomings of agents. We instead locate where the loop stays open: agents close it at the execution level but not at the strategy level, which motivates the strategy-level metrics in Section 5.

7 Conclusion and Limitation

This paper separates two capabilities that discussions of AI post-training AI tend to conflate: *executing* a committed strategy and *revising* it as evidence accumulates. Our trajectory analysis shows that frontier agents are reliable executors of LLM post-training, yet each locks into an agent-specific default strategy before any experimental evidence arrives. Testing whether the agent lacks experience, reasoning, or the decision to switch, we find that (1) experience improves execution but leaves the strategy unchanged; (2) additional reasoning compute refines the committed strategy rather than revising it; and (3) a human review before training changes *which* strategy the agent locks into, not *whether* it locks in, whereas a single mid-run instruction to switch outperforms the agent's own continuation under the same budget. **What is missing is thus not execution-level capability, but the decision to reopen a committed strategy and try another one.** The post-training loop of current agents closes only at the execution level and stays open at the strategy level. Closing it calls for interaction protocols and training signals that make strategy revision an explicit, rewarded decision.

Limitations. LLM post-training is a long-horizon, high-variance task with many interacting factors, and three independent cannot cover the full range of agent behavior, such as the unstable performance of Fable-5 (Appendix E.3). The controlled experiments use a single base model (Qwen3-1.7B-Base), three benchmarks, and a 10-hour budget on four A800 GPUs. Each mid-run comparison forks a randomly selected trajectory per benchmark, with the branch point and alternative strategy chosen by the authors; it shows that a better strategy was reachable, not which strategy is best.

8 AI USE STATEMENT

In this work, LLM agents are the object of study: the analyzed trajectories and trained checkpoints are produced by the agents themselves and are reported as experimental results. As authors, we used generative AI tools for tasks requiring disclosure: producing initial annotation labels for trajectory analysis, distilling open-source documentation into the skill library, and assisting with implementing the framework and analysis code. We also used generative AI tools for drafting and polishing text, creating figures, and formatting references. We have reviewed all AI-assisted work: all labels were manually audited, and all code was verified and tested by the authors. We take responsibility for the final content of this work.

9 REPRODUCIBILITY STATEMENT

The main text and appendix document the trajectory corpus, annotation protocol, and metric definitions, as well as the full setup of the controlled experiments, the experience-driven framework, the human plan-review transcripts, and the strategy-revision protocol. Our code, skill library, and analysis pipeline are released anonymously at the the anonymous repository linked in the abstract.

REFERENCES

Mersad Abbasi. What we learned from letting AI PostTrain AI. Thoughtful, April 2026. URL https://www.thoughtfullab.com/letting-ai-posttrain-ai.html.

Luke Bailey, Kaiyue Wen, Kefan Dong, Tatsunori Hashimoto, and Tengyu Ma. Scaling self-play with self-guidance, 2026. URL https://arxiv.org/abs/2604.20209.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing ai agent failures from execution trajectories, 2026. URL https://arxiv.org/abs/2602.02475.

Harshit Bisht, Vinay Kumar, Kevin Maik Jablonka, Mausam, and N. M. Anoop Krishnan. Agentic ai scientists are not built for autonomous scientific discovery, 2026. URL https://arxiv.org/abs/2605.08956.

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

Bin Chen, Shouliang Zhu, Beidan Liu, Yong Zhao, Tianle Pu, Huichun Li, and Zhengqiu Zhu. A2dept: Large language model-driven automated algorithm design via evolutionary program trees. *arXiv preprint arXiv:2604.24043*, 2026a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wanyi Chen, Xiao Yang, Xu Yang, Tianming Sha, Qizheng Li, Zhuo Wang, Bowen Xian, Fang Kong, Weiqing Liu, and Jiang Bian. Agentˆ 2 rl-bench: Can llm agents engineer agentic rl post-training? *arXiv preprint arXiv:2604.10547*, 2026b.

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, 2026. URL https://arxiv.org/abs/2608.20318.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Chao Deng, Shaolei Zhang, Ju Fan, and Xiaoyong Du. Dataevolver: Automatic data preparation for large language models through multi-level self-evolving. *arXiv preprint arXiv:2606.07001*, 2026.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, Zichen Liang, Boyuan Sun, Tianshuo Peng, Yifan Zhou, Xin Li, et al. Mlevolve: A self-evolving framework for automated machine learning algorithm discovery. *arXiv preprint arXiv:2606.06473*, 2026a.

Yaxin Du, Xiyuan Yang, Zhifan Zhou, Wanxu Liu, Zixing Lei, Zimeng Chen, Fenyi Liu, Haotian Wu, Yuzhu Cai, Zexi Liu, et al. Datamaster: Data-centric autonomous ai research. *arXiv preprint arXiv:2605.10906*, 2026b.

Zhiyuan Fan, Wenwei Jin, Feng Zhang, Bin Li, Yihong Dong, Yao Hu, and Jiawei Li. Evolving-rl: End-to-end optimization of experience-driven self-evolving capability within agents, 2026. URL https://arxiv.org/abs/2605.10663.

Haoyang Fang, Wei Zhu, Boran Han, Alex Zhang, Zhenyu Pan, Shuo Yang, Shuai Zhang, Jiading Gai, Peng Tang, Cuixiong Hu, et al. Llmzero: Discovering adaptive training strategies for rl post-training via llm agents. *arXiv preprint arXiv:2606.18388*, 2026.

Gabriel R Gonzalez, Johannes Habel, and Gary K Hunter. Ai agents, agentic ai, and the future of sales. *Journal of Business Research*, 202:115799, 2026.

Honglin Guo, Tao Gui, Kun Cai, Haodong Chen, Yicheng Chen, Guanting Dong, Qiming Ge, Yuyang Hu, Zixian Huang, Jiajie Jin, Alexander Lam, Yining Li, Jiahang Lin, Yanjiang Liu, Xinyu Lu, Haijun Lv, Zerun Ma, Junlin Shang, Qisheng Su, Guoqiang Wang, Rui Wang, Zhecan Wang, Hao Xiang, Xinchen Xie, Shuhao Xing, Xiaoyu Xing, Wanghan Xu, Xinyu Yang, Yajie Yang, Chengfeng Zhao, Haoran Zhao, Penghao Zhao, Ruojun Zhou, Yunhua Zhou, Dongsheng Zhu, Yicheng Zou, Qiye Cai, Xinmeng Che, Jiabei Chen, Jiahao Chen, Jiayi Chen, Yujia Chen, Lizhi Cui, Youheng Dai, Xin Deng, Yi Dong, Shihan Dou, Chenya Gu, Xu Guo, Ding Han, Feiyang Hao, Haotan He, Jie Hou, Binze Hu, Zijian Hu, Junhao Huang, Huicheng Jiang, Jiazhen Jiang, Shufan Jiang, Jiahao Kuang, Bowen Lai, Bo Li, Jiaqiang Li, Peng Li, Qilong Li, Zhuoqun Li, Jiaxiang Liu, Shuainan Liu, Tong Liu, Yi Liu, Zhonghang Lu, Jianwen Luo, Yanyi Luo, Huijie Lv, Ningsheng Ma, Houcheng Min, Chengjun Pan, Qiyuan Peng, Xiaoxuan Peng, Jianmin Qian, Jiantao Qiu, Wanying Ren, Huayu Sha, Jifei Shan, Zixin Shang, Bing Shao, Zhuohui Sheng, Jiayang Shi, Yang Shu, Aierpanjiang Simayi, Sirui Song, Yuxiao Song, Zhe Sun, Zhichao Sun, Wenzhe Tan, Wenhui Tian, Zhongbo Tian, Hanchen Wang, Pengbo Wang, Rui Wang, Yiding Wang, Yuhui Wang, Zhiheng Xi, Caijun Xu, Chao Xu, Yongfeng Xu, Xiaolei Yang, Zhixiong Yang, Qian Yao, Shihong Yi, Yuankai Ying, Jia Yu, Dingbo Yuan, Hao Yuan, Junjie Yuan, Bo Zhang, Caixian Zhang, Qiuyinzhe Zhang, Jiyuan Zhao, Ying Zhao, Pujun Zheng, Xiaoxue Zhong, Xiaohao Zhou, Xinyu Zhou, Guanru Zhu, Yulun Zhu, Yaojie Lu, Tao Ji, Hongyu Lin, Yutao Zhu, Pengfei Cao, Guoxiu He, Xianpei Han, Ben He, Zhicheng Dou, Kang Liu, Qi Zhang, Le Sun, Jun Zhao, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Bowen Zhou. Atria dawn: The dawn of agentic superintelligence, 2026a. URL https://arxiv.org/abs/2609.15818.

Taicheng Guo, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Autollmresearch: Training research agents for automating llm experiment configuration-learning from cheap, optimizing expensive. *arXiv preprint arXiv:2605.11518*, 2026b.

Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Andrej Karpathy. autoresearch: Ai agents running research on single-gpu nanochat training automatically. https://github.com/karpathy/autoresearch, 2026.

Naveen Krishnan. Ai agents: Evolution, architecture, and real-world applications. *arXiv preprint arXiv:2503.12687*, 2025.

Ilia Kulikov, Chenxi Whitehouse, Tianhao Wu, Yixin Nie, Swarnadeep Saha, Eryk Helenowski, Weizhe Yuan, Olga Golovneva, Jack Lanchantin, Yoram Bachrach, et al. Autodata: An agentic data scientist to create high quality synthetic data. *arXiv preprint arXiv:2606.25996*, 2026.

Qizheng Li, Yifei Zhang, Xiao Yang, Xu Yang, Zhuo Wang, Weiqing Liu, and Jiang Bian. Ft-dojo: Towards autonomous llm fine-tuning with language agents, 2026a. URL https://arxiv.org/abs/2603.01712.

Shuyue Stella Li, Rui Xin, Teng Xiao, Yike Wang, Rulin Shao, Zoey Hao, Melanie Sclar, Sewoong Oh, Faeze Brahman, Pang Wei Koh, and Yulia Tsvetkov. Evolm: Self-evolving language models through co-evolved discriminative rubrics, 2026b. URL https://arxiv.org/abs/2605.03871.

Yu Li, Chenyang Shao, Xinyang Liu, Ruotong Zhao, Peijie Liu, Hongyuan Su, Zhibin Chen, Qinglong Yang, Anjie Xu, Yi Fang, et al. Autosota: An end-to-end automated research system for state-of-the-art ai model discovery. *arXiv preprint arXiv:2604.05550*, 2026c.

Honghao Lin, David P. Woodruff, Yuan Deng, Jieming Mao, Song Zuo, and Vahab Mirrokni. Stellar colosseum: A many-agent harness for long-horizon research in mathematics and theoretical computer science, 2026. URL https://arxiv.org/abs/2609.15983.

Kaiyuan Liu, Qiuyang Mang, Bo Peng, Wenhao Chai, Hanchen Li, Shreyas Pimpalgaonkar, Luke Zettlemoyer, Alex Dimakis, and Alvin Cheung. When agents slow down: Understanding llm agents' test-time strategies via elo-per-token analysis, 2026. URL https://arxiv.org/abs/2609.15309.

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

Xinyu Lu, Tianshu Wang, Pengbo Wang, zujie wen, Zhiqiang Zhang, Jun Zhou, Boxi Cao, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. The meta-agent challenge: Are current agents capable of autonomous agent development?, 2026b. URL https://arxiv.org/abs/2606.04455.

Zeyuan Ma, Hongshu Guo, Yue-Jiao Gong, Jun Zhang, and Kay Chen Tan. Toward automated algorithm design: A survey and practical guide to meta-black-box-optimization. *IEEE Transactions on Evolutionary Computation*, 2025.

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, 2026. URL https://arxiv.org/abs/2607.25886.

NVIDIA NeMo Team. Nemo rl: A scalable and efficient post-training library. https://github.com/NVIDIA-NeMo/RL, 2025. GitHub repository.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. Gdpval: Evaluating ai model performance on real-world economically valuable tasks, 2025. URL https://arxiv.org/abs/2510.04374.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. Posttrainbench: Can llm agents automate llm post-training? *arXiv preprint arXiv:2603.08640*, 2026.

Bowen Ren, Heyan Huang, Yinghao Li, and Yang Gao. Metaevo: A meta-optimization framework for experience-driven agent evolution. *arXiv preprint arXiv:2606.07603*, 2026.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Chan Jun Shern, 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 *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, et al. Agents' last exam. *arXiv preprint arXiv:2606.05405*, 2026.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. Ai-researcher: Autonomous scientific innovation. *Advances in Neural Information Processing Systems*, 38:9481–9520, 2026.

Dhruv Trehan and Paras Chopra. Why llms aren't scientists yet: Lessons from four autonomous research attempts. *arXiv preprint arXiv:2601.03315*, 2026.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zhen Li, Xianfeng Tang, Qi He, et al. Agenttts: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. *Advances in Neural Information Processing Systems*, 38:98396–98433, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Jinhang Xu, Qiyuan Zhu, Yujun Wu, Zirui Wang, Dongxu Zhang, Marcia Tian, Yiling Duan, Siyuan Li, Jingxuan Wei, Sirui Han, Yike Guo, Odin Zhang, Conghui He, and Cheng Tan. Nanoresearch: Co-evolving skills, memory, and policy for personalized research automation. *arXiv preprint arXiv:2605.10813*, 2026a.

Weixian Xu, Tiantian Mi, Yixiu Liu, Yang Nan, Zhimeng Zhou, Lyumanshan Ye, Lin Zhang, Yu Qiao, and Pengfei Liu. Asi-evolve: Ai accelerates ai. *arXiv preprint arXiv:2603.29640*, 2026b.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, et al. R&d-agent: An llm-agent framework towards autonomous data science. *arXiv preprint arXiv:2505.14738*, 2025b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

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

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents, 2026b. URL https://arxiv.org/abs/2505.22954.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2026c.

Qifan Zhang, Dongyang Ma, Tianqing Fang, Jia Li, Jing Tang, Nuo Chen, Haitao Mi, and Yan Wang. Training llm agents for spontaneous, reward-free self-evolution via world knowledge exploration, 2026d. URL https://arxiv.org/abs/2604.18131.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, et al. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv:2604.08224*, 2026.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for llm agents, 2025a. URL https://arxiv.org/abs/2506.12928.

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

A Diagnosing Post-Training through Agent Trajectories

A.1 Overview

We diagnose how agents execute post-training directly from their execution trajectories (Barke et al., 2026). For each agent trajectory $\tau = \{(s_t, x_t, r_t)\}_{t=1}^{T}$, we reconstruct tool calls, shell commands, file edits, command outputs, training jobs, and evaluation results from the raw logs. We then annotate each training experiment with its strategy state $s_t \in \mathcal{S}$ and classify each pair of adjacent experiments as either a strategy change or an execution-level adjustment. Labels are produced by rule-based scripts together with an LLM, and the authors review all labels afterwards.

We analyze 1,338 trajectories from PostTrainBench (Rank et al., 2026), covering seven benchmarks (AIME 2025, ArenaHardWriting, BFCL, GPQA Main, GSM8K, HealthBench, and HumanEval),

four base models (Gemma-3-4B-PT, Qwen3-1.7B-Base, Qwen3-4B-Base, and SmolLM3-3B-Base), and 20 agent-model configurations spanning three agent frameworks (Claude Code, Codex CLI, and OpenCode). Each trajectory is produced under a 10-hour budget on one NVIDIA H100 80GB GPU.

A.2 Annotation Protocol

Training experiments. An experiment $(s_t, x_t, r_t) \in \tau$ is counted only when an executed command starts a model parameter update. Writing a script, preparing data, installing packages, evaluating a checkpoint, or merging an adapter does not count.

Strategy state $s_t \in \mathcal{S}$. For each verified experiment, we annotate the strategy state $s_t = (p, d, g) \in \mathcal{S} = \mathcal{P} \times \mathcal{D} \times \mathcal{G}$: the training algorithm p, the data source d, and the training stages g. The three components are annotated from separate evidence: p from the executed trainer and loss, d from file provenance and the commands that create the data, and g from checkpoint initialization. Full-parameter SFT and parameter-efficient fine-tuning (PEFT) are treated as distinct algorithms. Table 3 lists the algorithm labels.

Table 3: Training-algorithm labels and identification criteria.

Algorithm	Executed evidence
Full-parameter SFT	Likelihood training that updates all model parameters.
Parameter-efficient fine-tuning	Likelihood training through Low-Rank Adaptation (LoRA), quantized LoRA (QLoRA), or another adapter.
Reinforcement learning	Sampled model outputs optimized with a reward signal, including Proximal Policy Optimization (PPO) and Group Relative Policy Optimization (GRPO).
Preference optimization	Preferred/rejected response training, including Direct Preference Optimization (DPO)-style objectives.
Distillation	Training on supervision or outputs produced by a teacher model.

Strategy changes vs. execution changes. A transition between adjacent experiments is classified as a *strategy change* when $s_{t+1} \neq s_t$ (at least one recognized component of the strategy state changes). All other transitions—learning-rate tuning, reward shaping within the same algorithm, data formatting, checkpoint selection, and implementation repair—are classified as *execution-level adjustments* ($s_{t+1} = s_t$, $x_{t+1} \neq x_t$).

Training algorithm. Formally, the t-th experiment of trajectory i optimizes

$$J_{i,t}(\theta) = \mathbb{E}_{z \sim D_{i,t}} \left[\ell_{p_{i,t}}(\theta; z, \lambda_{i,t}) \right],$$

where $D_{i,t}$ is the training distribution, $\lambda_{i,t}$ denotes the remaining hyperparameters, and $\ell_{p_{i,t}}$ specifies the recognized training algorithm $p_{i,t}$.

Data source. Training data is labeled *curated*, *self-generated*, or *mixed* from file provenance and the commands that create it: a data file produced by a generation command (e.g., vLLM sampling that explicitly writes the file) is marked self-generated, a file produced by a hub download or format conversion is marked curated, and training data that combines both is marked mixed.

Strategy-change counts. Among the $3{,}557$ recognized adjacent pairs (Appendix A.3), we find 74 strategy changes (2.1%): 35 algorithm changes, 38 data-source changes, and 1 change in training stages. No pair changes more than one dimension (Tables 4 and 5).

Human validation. The authors review all labels produced by the scripts and the LLM; strategy-state labels are further checked against the executed trainer, loss, and retained evidence strings.

Initial strategy. The initial strategy is recognized for 814 of the $1{,}338$ trajectories, including trajectories that never launch training but state a planned strategy. Claude Code starts with full SFT in $166/231$ recognized cases (71.9%), whereas Codex CLI starts with PEFT in $274/306$ (89.5%). The

Table 4: Cross-framework strategy comparison per benchmark, pooled over the four base models. Final metrics are means over trained trajectories with a valid final score.

Agent framework	Benchmark	Final metric (n)	Initial strategy	Strategy changes
Claude Code	AIME 2025	0.049 (33)	Full SFT: 19/31 (61.3%)	14/132 (10.6%)
	ArenaHardWriting	0.135 (34)	Full SFT: 27/30 (90.0%)	19/174 (10.9%)
	BFCL	0.860 (44)	Full SFT: 33/51 (64.7%)	0/171 (0.0%)
	GPQA Main	0.279 (36)	Full SFT: 17/27 (63.0%)	4/171 (2.3%)
	GSM8K	0.556 (38)	Full SFT: 24/31 (77.4%)	8/187 (4.3%)
	HealthBench	0.281 (36)	Full SFT: 22/28 (78.6%)	1/179 (0.6%)
	HumanEval	0.480 (40)	Full SFT: 24/33 (72.7%)	8/189 (4.2%)
	Overall	–	Full SFT: 166/231 (71.9%)	54/1203 (4.5%)
Codex CLI	AIME 2025	0.009 (43)	PEFT: 40/42 (95.2%)	3/120 (2.5%)
	ArenaHardWriting	0.109 (39)	PEFT: 30/40 (75.0%)	11/159 (6.9%)
	BFCL	0.573 (44)	PEFT: 44/45 (97.8%)	0/67 (0.0%)
	GPQA Main	0.277 (46)	PEFT: 42/46 (91.3%)	0/152 (0.0%)
	GSM8K	0.443 (47)	PEFT: 36/43 (83.7%)	0/198 (0.0%)
	HealthBench	0.231 (42)	PEFT: 37/43 (86.0%)	0/106 (0.0%)
	HumanEval	0.332 (47)	PEFT: 45/47 (95.7%)	1/141 (0.7%)
	Overall	–	PEFT: 274/306 (89.5%)	15/943 (1.6%)
OpenCode	AIME 2025	0.018 (38)	Full SFT: 25/41 (61.0%)	1/196 (0.5%)
	ArenaHardWriting	0.091 (34)	Full SFT: 27/39 (69.2%)	1/235 (0.4%)
	BFCL	0.372 (28)	Full SFT: 23/36 (63.9%)	0/195 (0.0%)
	GPQA Main	0.172 (36)	Full SFT: 25/42 (59.5%)	0/213 (0.0%)
	GSM8K	0.381 (36)	Full SFT: 24/40 (60.0%)	2/219 (0.9%)
	HealthBench	0.219 (35)	Full SFT: 29/40 (72.5%)	1/159 (0.6%)
	HumanEval	0.263 (36)	Full SFT: 31/39 (79.5%)	0/194 (0.0%)
	Overall	–	Full SFT: 184/277 (66.4%)	5/1411 (0.4%)

Table 5: Decomposition of strategy changes among the 3,557 recognized adjacent experiment pairs.

Changed dimension	Pairs	Rate
Training algorithm	35	0.98%
Data source	38	1.07%
Training stages	1	0.03%
Overall	74	2.08%

same direction holds in every combination of the seven benchmarks and four base models: Claude Code has a higher full-SFT share and Codex CLI a higher PEFT share (per-benchmark shares in Table 4).

A.3 Statistics

Of the 1,338 trajectories, 900 launch at least one model parameter update, yielding 5,111 verified training experiments in total. Among these, we recognize the training algorithm for 4,378 experiments from the executed trainer and loss; the other 733 remain unlabeled. Within each trajectory, the labeled experiments are chained in execution order, bridging over unlabeled experiments (e.g., an SFT $\rightarrow$ unknown $\rightarrow$ GRPO sequence yields one SFT $\rightarrow$ GRPO pair). Each consecutive pair in this chain is a *recognized training pair*. All switch rates—the pooled $\bar{\rho} = 74/3{,}557$ (2.1%), the per-agent ρ_a of Table 1, and the per-benchmark rates of Table 4—are computed over these pairs.

The concentrations κ_a of Table 1 are computed over the 814 recognized initial strategies (231, 306, and 277 for Claude Code, Codex CLI, and OpenCode, respectively).

The data-source dimension applies to the 4,344 experiments with a supervised objective. Data provenance is identifiable for 1,801 of them (1,327 curated, 424 self-generated, 50 mixed), spanning 1,401 recognized training pairs.

A.4 AUDIT OF TRAINING-ALGORITHM CHANGES

We audit the 16 trajectories with a training-algorithm change because those changes admit the clearest before–after comparisons. Fourteen begin with SFT, 11 switch at least twice, and 15 of the 35 transitions return to SFT. The first switch occurs at a median normalized experiment progress of 0.40, and the median over all switches is 0.67.

Table 6 shows both positive and negative cases: an alternative algorithm sometimes yields a gain and sometimes regresses.

Table 6: Trace-level case studies among the 16 trajectories with a training-algorithm change. The reported values come from the trajectory logs and retain each log's original comparator and sample size.

Case	Setting	Change	Observation	Interpretation
3fd3ea0b	ArenaHardWriting, SmolLM3-3B, Codex CLI	SFT → DPO	Held-out 256-pair proxy: 66.0% → 66.8%	Positive; $+0.8$ pp
c92715d6	ArenaHardWriting, Gemma-3-4B, Claude Code	SFT → DPO	Benchmark win rate: $\sim 8\%$ → 16.7%	Positive; approximately $2\times$
50c64287	ArenaHardWriting, Qwen3-4B, Codex CLI	SFT → DPO	Same 8-prompt local proxy: 7.14% → 7.69%	Positive; small sample
860ceacd	AIME 2025, Qwen3-1.7B, Claude Code	SFT → GRPO	30-problem evaluation: 0% → 3.3% (1/30)	Positive; high variance
8226f786	ArenaHardWriting, SmolLM3-3B, Codex CLI	SFT → DPO	External proxy: 0.539 → 0.478; internal DPO reward accuracy: 0.713 → 0.727	Counterexample; metric mismatch

B CONTROLLED EXPERIMENT PROTOCOL

Setup. The controlled experiments use Qwen3-1.7B-Base on GSM8K, HumanEval, and AIME 2025. The autonomous baselines are Claude Code powered by Opus 4.6 and by GLM-5.2, and Codex CLI powered by GPT-5.2. All interventions use Claude Code with Opus 4.6. Each configuration has three independent 10-hour runs on four NVIDIA A800 GPUs. Within a comparison, the base model, benchmark, hardware budget, system prompt, and evaluator are fixed. Human guidance (Section 4.3) builds on the full experience-driven framework and adds either a human review before training or a single mid-run instruction to switch strategy.

Evaluation. GSM8K and HumanEval use pass@1 accuracy. AIME 2025 contains only 30 problems, so one solved problem changes pass@1 by 3.33 percentage points. We therefore evaluate submitted checkpoints with pass@8, using eight completions per problem and the same evaluator and decoding configuration across conditions.

Metric Conventions in Table 2. GSM8K and HumanEval report pass@1 on the full official test sets; the AIME 2025 column reports the pass@8 protocol above, applied to the submitted checkpoints. Each entry is the mean over the three runs' submitted checkpoints with one standard deviation across runs. The parenthesized deltas are differences of these means: baseline and experience-driven rows are compared against the base model, and each ablation row against the full experience-driven framework. The official instruct model row uses the instruct counterpart of Qwen3-1.7B-Base, evaluated with the same scripts. *Gap Closed* is $(\mathrm{Avg} - \mathrm{Avg}_{\mathrm{base}})/(\mathrm{Avg}_{\mathrm{instruct}} - \mathrm{Avg}_{\mathrm{base}})$, the fraction of the base-to-instruct gap in average score recovered by each setting.

Trajectory Phases. For Figure 3, one interaction turn consists of an assistant message and its tool calls and results. Each turn is assigned one of eight labels: *Exploration*, *Data/Setup*, *SFT*, *RL/GRPO*, *Evaluation*, *Debugging*, *Journal/Skill*, or *Checkpoint/Waiting*. An LLM assigns the initial label and the authors review it. Time between consecutive turns is attributed to the phase that launched the operation.

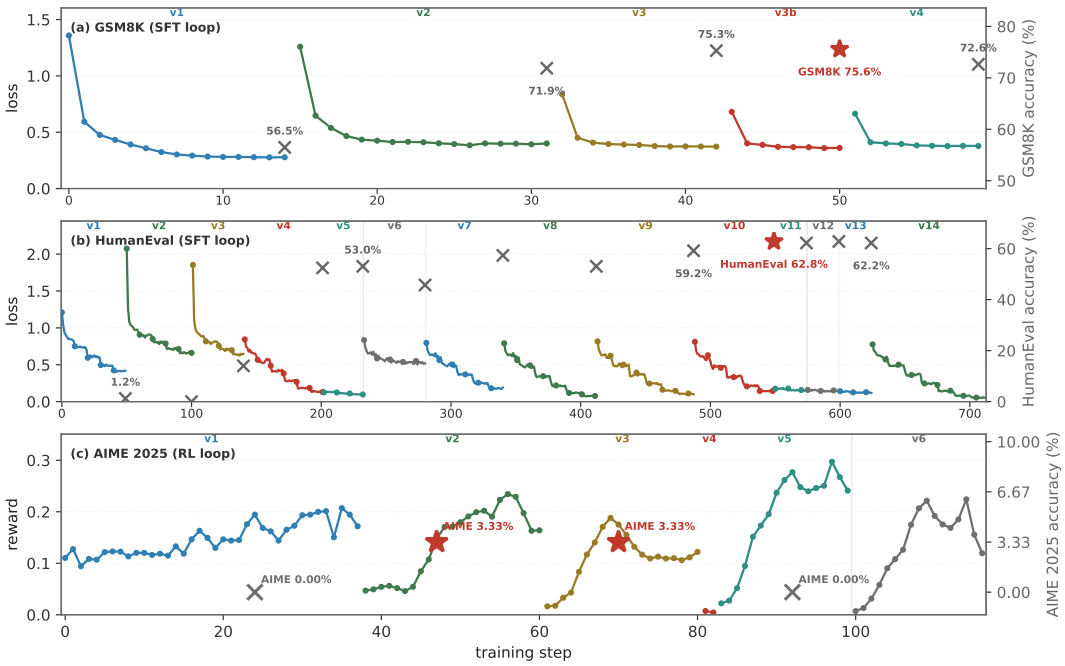


Figure 8: Training dynamics under the experience-driven framework (best-performing run). Later experiments plateau or regress, while subsequent actions remain within the committed strategy.

C Experience-Driven Framework

The framework supplies three components while leaving all training decisions to the main agent. This appendix details the construction and observed usage of each component.

C.1 Experiment Journal

Long-horizon post-training generates a large amount of task-specific information, and as the agent context grows, observations from earlier runs become difficult to track. The experiment journal addresses this with a structured, append-only log with typed entries: `plan`, `observation`, `lesson`, `eval_result`, and `eval_analysis`. Representative entries show that the journal captures pipeline-relevant conclusions even when they do not produce a strategy revision. For example, on HumanEval, the agent records "*SFT plateau confirmed at 102–103 [of 164] for continuation.*" The evaluator writes "*ABANDON FURTHER SFT: v5 proves diminishing returns . . . further SFT iterations will show similar marginal gains.*" The run nevertheless continues with additional SFT variants. On AIME 2025, the journal records "*entropy coefficient is extremely sensitive . . . the only stable setting is 0.003*" after three consecutive aborted retries of the same local adjustment.

C.2 Skill Library

We construct the skill library by distilling raw documents from open-source projects into a compact set of skills. First, we collect 908 documents (approximately 937K words) from the framework documentation, training recipes, and issue threads of projects such as verl (Sheng et al., 2024), TRL (von Werra et al., 2020), OpenRLHF (Hu et al., 2024), NeMo-RL (NVIDIA NeMo Team, 2025), and slime (Zhu et al., 2025b). Second, we distill this material into a 60-page knowledge wiki (approximately 20K words) containing project summaries, algorithm and framework entities, concepts, and synthesized experience guides. Third, we compress the wiki into `SKILL.md` files (approximately 1.2K words per file). After filtering, the seed skills include `posttraining-known-pitfalls`, `diagnose-silent-rl-failures`, `data-parquet-schema`, `monitor-rl-training`, and `create-skill`. They cover

crash and silent-failure diagnosis, generic RL triage, data formatting, runtime health, and the persistence of new experience.

On average, the agent consults skills 18 times on GSM8K, 20 times on HumanEval, and 60 times on AIME 2025 per run. In the AIME 2025 runs with human guidance at the initial decision (Section 4.3.1), it consults them only about twice per run. Despite this read activity and the presence of the `create-skill` meta-skill, **the agent creates no new skills in any run**. This asymmetry between consuming existing knowledge and producing reusable knowledge parallels the adoption gap between execution-level and strategy-level suggestions (Section 4.1.2).

C.3 Evaluator Agent

The evaluator agent uses the same configuration as the main agent (Claude Code with Opus 4.6). When the main agent requests an evaluation, the evaluator agent inspects the current pipeline and accumulated evidence, forms expectations about the checkpoint's behavior, invokes the original evaluation script, and analyzes both the scores and the model outputs. It returns concrete diagnoses and actionable suggestions, which may include identifying an implementation failure, proposing data or hyperparameter changes, or proposing an alternative training strategy. In this way, the evaluator agent absorbs high-volume, low-signal observations and returns compact, decision-relevant diagnoses to the main agent.

We extract the evaluator's recommendations from one representative experience-driven trajectory per benchmark. Table 7 separates execution-level from strategy-level suggestions.

Suggestion Extraction and Classification. We parse the evaluator's analysis entries recorded in the experiment journal of each selected run and group the recommendations into recurring suggestion themes; each occurrence of a theme in an evaluation cycle counts as one suggestion, since the main agent has a fresh opportunity to adopt it in every cycle. A suggestion counts as *adopted* when a subsequent experiment implements it. A suggestion is *strategy-level* if it departs from the committed strategy—switching the training algorithm (e.g., SFT to RL with a code-execution reward), adding or removing a training stage (e.g., an SFT warm-up); advice on rewards, hyperparameters, data construction, formatting, or checkpoint management within the committed strategy is execution-level. In none of these runs does the main agent launch a training run with a strategy other than the committed one.

Table 7: Evaluator suggestions and their adoption (adopted/total). Counts are theme occurrences per evaluation cycle and match Figure 4.

Benchmark	Eval cycles	Execution sugg. (adopted)	Strategy sugg. (adopted)
GSM8K	11	6/6	0/7
HumanEval	14	11/11	0/11
AIME 2025	5	5/5	0/3
Total	30	22/22	0/21

On HumanEval, 11 of the 14 evaluation cycles recommend RL. The main agent writes GRPO scripts but launches 14 SFT variants and no RL training. On AIME 2025, the evaluator proposes an SFT warm-up three times, but the agent continues GRPO-only training (Figure 9).

C.4 Related Work on Experience-Driven Agents

Experience-driven systems improve performance through memory, reusable skills, feedback, and externalized knowledge. Foundational approaches include verbal reflections (Shinn et al., 2023), natural-language lessons (Zhao et al., 2024), executable skill libraries (Wang et al., 2023), runtime experience (Zhou et al., 2026), and lifelong learning (Ma et al., 2025). Recent work extends these ideas to co-evolving skills, memory, and policy (Xu et al., 2026a); self-improving harnesses (Zhang et al., 2026a); meta-learning from experience (Ren et al., 2026; Fan et al., 2026); and learning from early exploratory trajectories (Zhang et al., 2026c). Related approaches also use self-generated feedback, co-evolving rubrics, self-play, self-distillation, and open-ended code evolution (Li et al.,

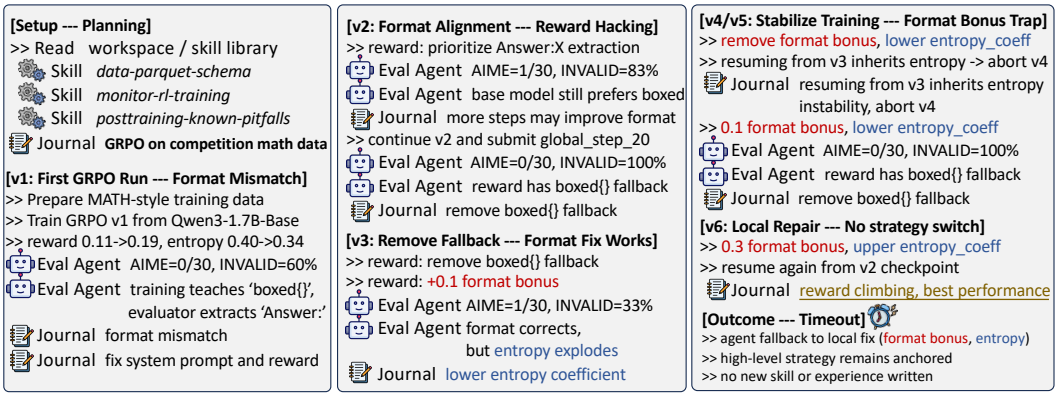


Figure 9: An annotated experience-driven trajectory on AIME 2025. The agent consults skills, maintains the experiment journal, and receives evaluator diagnoses throughout six training versions. All revisions remain at the execution level.

2026b; Bailey et al., 2026; Zhang et al., 2026d; Shenfeld et al., 2026; Zhang et al., 2026b). We adapt these mechanisms to LLM post-training to test whether supplying experience leads the agent to revise its strategy.

D Human Guidance at Critical Decision Points

This appendix details human guidance at the two decision points of Section 4.3: a human review at the initial choice of strategy, and a single instruction to switch strategy mid-run.

D.1 Human Guidance at the Initial Decision

D.1.1 Human Review

The runs with human guidance on AIME 2025 (Section 4.3.1) go through multiple review iterations before training begins. In each iteration, the agent proposes its strategy as a complete research plan, and the reviewer returns a fixed-format JSON decision that either approves the plan or requests a revision with an explicit rationale. The transcript below is from a representative run.

Iteration 1 (decision: revise). The agent first proposes an SFT pipeline (approximately 10K–30K examples over 2–3 epochs). The reviewer requests a revision:

> "SFT should be a minimal formatting warm-up only, stop once the model can reliably produce the required answer format and switch to RL"

Iteration 2 (decision: approve). The revised plan reduces SFT to a formatting-only warm-up (at most 1K–3K examples for one epoch, with an explicit reasoning-degradation check). It allocates the main budget to GRPO with large rollout groups and long generations. The reviewer approves the plan with one clarification:

> "Ensure that the evaluation format is aligned to the benchmark prompt."

Once the strategy is approved, the run proceeds fully autonomously. The agent inspects the base model, finds the required output format already attainable without SFT, and skips the warm-up altogether. It allocates the full training budget to GRPO.

D.1.2 Iteration Within the New Strategy

The runs with human guidance on AIME 2025 reach their best score within the first few training runs; later training runs mostly score zero and never recover the peak (Figure 6a). Concretely, in

these later training runs the agent tunes the entropy coefficient and learning rate when resuming from earlier checkpoints, but neither restores the best checkpoint nor treats the regression as a trigger to reconsider its current strategy.

D.2 Human Guidance at a Mid-Run Decision

This subsection details the mid-run forks of Section 4.3.2, which test whether a single instruction issued mid-run can reopen a strategy the agent has already committed to. For each benchmark, we fork a completed autonomous trajectory at a selected mid-run checkpoint. The agent's own recorded continuation follows the actions it actually took, whereas the guided branch receives a single instruction to switch to an alternative strategy. The branch points and alternative strategies were selected by the authors after the trajectories had completed, so the comparisons are controlled case studies. We compare the two under the same hardware and remaining budget.

D.2.1 Experimental Setup

We study three completed autonomous trajectories produced by Claude Code powered by Opus 4.6 on Qwen3-1.7B-Base: one each on GSM8K, HumanEval, and AIME 2025. The three trajectories are run under the experience-driven framework of Section 4.1 with no human involvement, on four NVIDIA A800 80GB GPUs with a 10-hour wall-clock budget, independently of the three-run sets of Table 2. At a selected branch point t_b, the recorded continuation follows the subsequent actions taken autonomously by the agent. The guided branch starts from the checkpoint and experimental state available at t_b with a single instruction to switch to a named alternative strategy; the agent itself carries out the subsequent implementation, training, and evaluation. All costs of the guided branch count toward the remaining budget, so its total elapsed time satisfies

$$T_{\text{guided}} \leq B_{t_b},$$

where B_{t_b} denotes the remaining budget at the branch point. We evaluate GSM8K and HumanEval with pass@1 and AIME 2025 with pass@8.

D.2.2 Overall Results

Figure 6b summarizes the results of the guided branches. On all three benchmarks, the guided branch reaches a higher final score than the agent's own recorded continuation. The gain is largest on GSM8K (17.44 points) and smaller on HumanEval (10.8 points), while on AIME 2025 the gain from $0/30$ to $2/30$ is a small-sample existence result.

GSM8K: switching from SFT to GRPO. In the GSM8K trajectory, the agent first uses full-parameter SFT to repair the output format and then continues with five additional SFT runs, varying the data version, learning rate, and number of epochs. The recorded continuation finally obtains 47.76% on the full GSM8K test set.

At the branch point after the first SFT run, the guided branch starts from the SFT checkpoint and switches to GRPO with DAPO-style modifications (Yu et al., 2025). It uses only the official GSM8K training set and a numerical answer-matching reward and reaches 65.20% within the remaining budget. This result shows that a better route than the recorded SFT continuation was reachable from the same checkpoint.

HumanEval: switching from SFT to training with execution feedback. In the HumanEval trajectory, the agent obtains an early improvement by aligning the supervised data with the function-completion format required by the benchmark. After this correction, the agent runs several SFT variants with different data versions, learning rates, training lengths, and initialization checkpoints. These runs mostly remain near a performance plateau, and the final checkpoint of the recorded continuation achieves 52.0% on the full benchmark.

The guided branch starts from the eighth checkpoint. It first constructs execution-verified self-generated data from MBPP and performs rejection-sampling fine-tuning (RFT), followed by online GRPO with unit-test rewards, which reaches 62.8% at step 240.

AIME 2025: switching from SFT to GRPO with verifiable rewards. The AIME 2025 trajectory initially focuses on instruction following, output formatting, and end-of-sequence behavior. After a 4-hour SFT stage, the model produces parseable mathematical answers but still solves none of the AIME 2025 problems. The agent then performs four additional SFT runs, changing the difficulty and composition of synthetic data, the learning rate, the training length, and the maximum sequence length; all of these runs remain at 0%.

The guided branch starts from the sixth checkpoint and switches to GRPO with a verifiable final-answer reward. It uses filtered GSM8K data together with DAPO-Math data (Yu et al., 2025) and completes 150 training steps in approximately 3.5 hours, within the remaining 4 hours and 56 minutes. On AIME 2025, the result improves from 0/30 to 2/30 under pass@8.

D.2.3 Forks with an Instruction to Reconsider

To rule out that the agent merely lacks a prompt to reconsider (Section 4.3.2), we additionally fork each of the three trajectories at the same branch point as its guided branch. Instead of naming an alternative, each fork begins with an instruction that asks the agent to reconsider its current strategy; the agent then plans, trains, and evaluates on its own under the same remaining budget. Table 8 compares the strategies and final scores of the recorded continuation, this fork, and the guided branch after the branch point.

The agent reaffirms its current strategy and adjusts its execution. In every fork, the agent keeps SFT and spends the remaining budget on execution-level changes: the data mixture, the format of the training targets, data filtering, and hyperparameters. No fork introduces a new objective or training stage, such as the RL stages of the guided branches, although all three benchmarks provide a verifiable reward. The only change in training algorithm, from full-parameter SFT to LoRA in the AIME 2025 fork, is driven by GPU availability rather than by the evidence: a previously launched training run has not yet finished, so the agent judges that the current GPUs cannot accommodate another full-parameter run and continues the same SFT with a LoRA adapter.

Final scores stay near or below the recorded continuations. No fork improves on its recorded continuation beyond evaluation variance. On GSM8K, the selected checkpoint scores 48.0% on a 150-problem subset, close to the 47.76% of the recorded continuation and far below the 65.20% of the guided branch. On HumanEval, the selected checkpoint scores 31.7% on the full benchmark, below the 52.0% of the recorded continuation, and the four training runs of the fork consume nearly half of the remaining budget. On AIME 2025, both SFT checkpoints of the fork solve 1 of 30 problems in the in-run pass@1 evaluation, within evaluation variance. A reminder to reconsider therefore does not substitute for the decision itself: without a named alternative, the agent reaffirms its current strategy.

Table 8: Strategies and final scores after the branch point in the agent's own recorded continuation, the fork with an instruction to reconsider (reconsider fork), and the guided branch. The recorded continuation and the reconsider fork both stay with SFT, whereas each guided branch moves to RL with GRPO and reaches the highest final score. Scores are accuracy (%) on GSM8K and HumanEval and solved problems out of 30 under pass@8 on AIME 2025.

	GSM8K		**HumanEval**		**AIME 2025**	
	Strategy	Score	Strategy	Score	Strategy	Score
Recorded continuation	SFT	47.76	SFT	52.0	SFT	0/30
Reconsider fork	SFT	48.0	SFT	31.7	SFT, LoRA	1/30
Guided branch	GRPO	**65.20**	RFT, GRPO	**62.8**	GRPO	**2/30**

E Cross-Agent Analysis

Setup. The first two subsections analyze six supplementary trajectories for each of GPT-5.2 (Codex CLI) and GLM-5.2 (Claude Code)—one on each of GSM8K, HumanEval, and AIME 2025 at two

model scales (Qwen3-1.7B-Base and Qwen3-4B-Base)—each run under a 10-hour budget on four NVIDIA A800 GPUs. These runs are independent of the controlled experiments of Section 4 and of the released corpus of Appendix A.3; they replicate the default-strategy divergence of Section 3.2 under a different resource budget. A training launch counts as *successful* if the training process completes and saves a usable checkpoint, and an evaluation is *valid* if the evaluation command completes and returns a parsable score. Tables 9 and 10 summarize each trajectory.

E.1 GPT-5.2 (CODEX CLI)

The default-strategy divergence replicates: Codex CLI searches predominantly within the adapter family. Across both model scales, 64 training commands complete successfully: 56 (87.5%) use PEFT and 8 (12.5%) use full SFT, with PEFT accounting for 28 of 35 successful 1.7B runs (80.0%) and 28 of 29 successful 4B runs (96.6%). The loop is compact: inspect the evaluation harness and chat template, align the output contract, construct SFT data, train a LoRA/QLoRA adapter or a full-SFT continuation, merge or export a candidate checkpoint, and run small- or medium-sample evaluations.

Feedback is frequent but mostly converted into execution-level changes. The trajectories contain 104 valid evaluations: 39 on GSM8K, 33 on AIME 2025, and 32 on HumanEval. The 1.7B runs improve GSM8K and moderately improve HumanEval; the 4B runs achieve a strong HumanEval result but reduce GSM8K below the base model; neither scale yields a reliable AIME improvement (Table 9). On the 1.7B trajectories with detailed wall-clock annotation, the combined 28.40 hours comprise 17.56 hours of training (61.8%), 3.67 hours of evaluation (12.9%), 1.03 hours of data processing (3.6%), 2.75 hours of inspection and debugging (9.7%), 0.74 hours of merge or export (2.6%), and 2.66 hours of idle or uncategorized gaps (9.4%); the AIME run spends 87.6% of wall time in training without a reliable gain. In summary, GPT-5.2 (Codex CLI) is effective at execution-level diagnosis—it identifies output-protocol mismatches, repairs end-of-sequence (EOS) and template problems, and repeatedly evaluates candidates—but negative evidence mostly leads to nearby data, template, hyperparameter, and checkpoint variants, and all 64 successful training runs remain within supervised fine-tuning.

Table 9: Trajectory-level summary of GPT-5.2 (Codex CLI) behavior at both model scales. Scores are in-run diagnostic pass@1 accuracies; "@ n" gives the number of evaluated problems, and "Final" denotes the checkpoint exported by the agent.

Model	Benchmark	Evals	Diagnostic result	Trajectory diagnosis
Qwen3-1.7B	GSM8K	20	Best/final 0.587 @ 150	LoRA and full-SFT continuations both benefit from prompt-contract, EOS, and answer-only repairs.
Qwen3-1.7B	HumanEval	14	Best 0.273 @ 150 / 0.262 @ 164	MBPP, CodeSearchNet, and template repairs help, but 20–50-sample evaluations overstate the full-benchmark quality.
Qwen3-1.7B	AIME 2025	22	Best 1/30; final repeatedly 0/30	Many math-data and LoRA/SFT variants produce no stable gain; 1/30 is within evaluation variance.
Qwen3-4B	GSM8K	19	Base 0.507 @ 150; final 0.467 @ 150	After 13 successful training runs and 12 merges, the selected model remains worse than the base model.
Qwen3-4B	HumanEval	18	Base 0.420 @ 150; final 0.687 @ 150	MBPP-only data, a no-<think> template, checkpoint selection, and sampling configuration recover from several zero-scoring early runs.
Qwen3-4B	AIME 2025	11	Best 1/30; final 0/30	Seven successful QLoRA runs fail to improve over the noisy base result; intermediate outputs show tokenizer and generation corruption.

E.2 GLM-5.2 (CLAUDE CODE)

Every successful launch uses full SFT; the training algorithm never changes. Across 32 training launches, 24 complete successfully, all with full SFT; the agent uses no LoRA, QLoRA, or adapter merge, and produces 51 valid evaluations (28 on GSM8K, 11 on AIME 2025, and 12 on HumanEval). The typical loop inspects the benchmark contract, builds a large task-specific SFT set, trains full weights, diagnoses stopping or generation failures, optionally generates rejection-filtered data, and exports a selected checkpoint (Table 10).

Data-source revision appears, but within a fixed algorithm. Under the criterion of Section 2, introducing rejection-filtered, self-generated data is a strategy change in the data-source dimension—the most common form of strategy change in the corpus (Table 5). GSM8K forms the most complete loop: three rejection-sampling rounds feed successive SFT stages, and repeated evaluations show that intermediate checkpoints outperform the final training steps. Outcomes are scale-dependent: the 1.7B trajectories (11 successful runs, 17 valid evaluations) improve GSM8K and HumanEval substantially but leave AIME 2025 at 0/30 even after combining rejection-filtered generations with SFT data, whereas the 4B trajectories (13 successful runs, 34 valid evaluations) improve all three benchmarks. On AIME 2025 at the 4B scale, the decisive intervention is execution-level: correcting an unintended 2,048-token cap in `generation_config.json` raises the two trained runs from 1/30 and 2/30 to 7/30. In summary, GLM-5.2 makes larger pipeline changes than GPT-5.2—rebuilding data, introducing rejection-filtered generations, and continuing from selected checkpoints—yet it never changes the training algorithm.

Table 10: Trajectory-level summary of GLM-5.2 (Claude Code) behavior at both model scales. Scores are in-run diagnostic pass@1 accuracies; notation as in Table 9.

Model	Benchmark	Evals	Diagnostic result	Trajectory diagnosis
Qwen3-1.7B	GSM8K	4	Base 0.200 @ 50; final 0.493 @ 150	Full SFT learns the answer contract; data cleaning and EOS repair further improve the result despite repeated out-of-memory failures.
Qwen3-1.7B	HumanEval	7	Base 0.116 @ 164; final 0.433 @ 164	Function-completion data and completion-only loss produce a large gain; a better run2 result is evaluated but not exported.
Qwen3-1.7B	AIME 2025	6	Base 0/5; final 0/30	Two full-SFT rounds, EOS repair, and rejection-filtered training complete successfully but yield no solved problem.
Qwen3-4B	GSM8K	24	Best 0.680 @ 200; final 0.654 @ 700	Three rejection-sampling rounds and four SFT stages improve the model; intermediate-checkpoint selection is consistently important.
Qwen3-4B	HumanEval	5	Base 0.300 @ 10; final 0.533 @ 150	Three SFT stages add code data and progressively lower the learning rate, improving the 150-sample result from 0.400 to 0.533.
Qwen3-4B	AIME 2025	5	Base 0/6; final 7/30	Correcting an unintended 2,048-token generation cap raises both trained runs to 7/30; run1 is retained as the final model.

E.3 FABLE 5 (CLAUDE CODE)

Beyond the Opus 4.6 used in our main experiments, we also analyze six preliminary runs of a more recent agent, Fable 5 (Claude Code), on Qwen3-1.7B-Base (Table 11). Fable 5 occasionally exceeds Opus 4.6: its strongest HumanEval run reaches 54.3% against 32.0% for Opus 4.6. However, several problems recur across its runs.

Performance is unstable. Some runs obtain a substantial gain after a few targeted corrections, while others spend most of the budget on infrastructure. In one AIME 2025 run, roughly 72% of the wall-clock budget goes to diagnosing training throughput, leaving about 83 minutes for training and

evaluation and producing no reliable gain. The strongest GSM8K and HumanEval trajectories, by contrast, improve after correcting output-format, stopping, or data-mixture problems.

Unintended behaviors appear. Some records involve data contamination and reward hacking, including modifications to the official evaluation interface. These behaviors inflate reported scores without improving the model.

Table 11: Trajectory-level summary of the Fable 5 (Claude Code) runs. Scores are reported with the evaluation protocol and sample size used in the supplementary logs.

Benchmark	Record	Base	Reported final result	Trajectory diagnosis
GSM8K	A	0.1243 (1319)	0.3490 ± 0.0119 (1319, three repeats)	Format-aligned data, assistant-only loss masking, and stop-token repair produce the clearest gain; continuation and checkpoint selection remain protocol-sensitive.
GSM8K	B	0.160 (50)	0.020 (full evaluation)	Training loss and token accuracy improve, but the full-evaluation score remains far below the 50-problem base estimate; the differing evaluation sizes limit the comparison.
HumanEval	C	0.060 (50)	0.2369 ± 0.0302 (164, seven repeats)	The base and final scores use different sample sizes; the run terminates after an API failure, so the result is useful mainly as evidence of evaluation variance.
HumanEval	D	0.060 (150)	0.5427 (164, official path)	Several continuations and data-mixture changes produce intermediate results from approximately 0.400 to 0.555; the largest gain follows a change in the proportion of verified self-distillation data.
AIME 2025	E	0.0333 (1/30)	0 (0/30, repeated)	No fine-tuned checkpoint reproduces or exceeds the single solved problem in the base evaluation; long-output and truncation failures dominate the final trajectory.
AIME 2025	F	0 (0/30)	0 official; 0.133 self-evaluation	The reported positive result comes from a non-official evaluation path and is not directly comparable with the official final score.

These behaviors fall outside our focus on the gap between execution-level and strategy-level capability; characterizing them, their causes, and their implications for evaluation design is left to future work.