# HARD ENOUGH TO LEARN: EVOLVING VERIFIED TOOL-USE ENVIRONMENTS WITH CODE AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large language model (LLM) agents have emerged as a dominant paradigm in artificial intelligence, with their capabilities increasingly determined by their ability to invoke and orchestrate external tools. Reinforcement learning acquires that ability at scale, and it needs verified environments with evolving difficulty to keep the training signal rich. We build a factory that produces verified RLVR data with code agents. An agent authors a domain, tools with implementations, a seeded database, and declarative task templates. Deterministic code then instantiates each task, derives its verified gold tool-call chain. One deterministic predicate validates the gold, measures each task's difficulty, and serves as the reward, so no judge sits in the loop. The factory remains effective across different code agents and harnesses. Per-task difficulty estimates calibrate the corpus to a learnable band and let the code agents harden the saturated tasks. Reinforcement learning on the band-calibrated corpus improves Qwen3.5 tool use across scales. We evaluate these capabilities on GAIA2, WorkBench, $\tau^3$-bench, BFCL, Workplace Assistant, and ToolSandbox, and assess capability retention on general benchmarks and IFBench. Training on this corpus outperforms training on randomly sampled task sets of equal and twice its size. The same corpus supports supervision beyond rewards. Supervised fine-tuning develops tool-use capabilities in weak non-Qwen base models from the Llama-3.2 and SmolLM families. We also demonstrate knowledge transfer through on-policy distillation within the generated environments. The 4B student achieves approximately 98% of the average performance of its 9B teacher across the evaluated benchmarks, while the 2B student outperforms a baseline trained with supervised fine-tuning followed by GRPO.

## 1 INTRODUCTION

Reinforcement learning for tool use runs on environments, and clean ones are hard to synthesize (Team, 2026b;a; Dong et al., 2026). Earlier tool-use synthesis focused on single-call API tasks; recent multi-turn pipelines (Prabhakar et al., 2025; Zeng et al., 2026) still rely on model-generated trajectories filtered by a language-model judge rather than on deterministic replay in a verified stateful environment, which leaves a reward a policy can game (Mahmoud et al., 2026; Wang et al., 2026a); and where an environment does exist, its tasks tend to arrive invalid or already saturated. A newer generation of evolutionary pipelines (Gao et al., 2026) adapts training data as the model improves, but typically relies on coarse success signals rather than task-level estimates of what remains learnable for the current student. NeoHorse-1 (Team et al., 2026) uses capability feedback to rebalance the training mixture, but does not use student outcomes to evolve the tasks themselves. We build verified tool-use environments with code agents. Code agents create stateful domains, tools, databases, and task templates. Deterministic execution instantiates each task, derives its gold tool-call chain, and admits it only after successful replay. The same deterministic verifier provides the training reward, with no language-model judge in the loop.

A verified task is not yet a useful one. On the base corpus a 4B student solves most tasks on every attempt, and eight identical successes give neither RL nor rejection-sampling FT a gradient. We measure each task's pass rate and build the corpus toward a learning band, the range where the student sometimes succeeds and sometimes fails: the code agents rewrite a saturated task's rules, and the generator re-derives and replays a harder gold from scratch. This identifies what makes tool-use tasks



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.

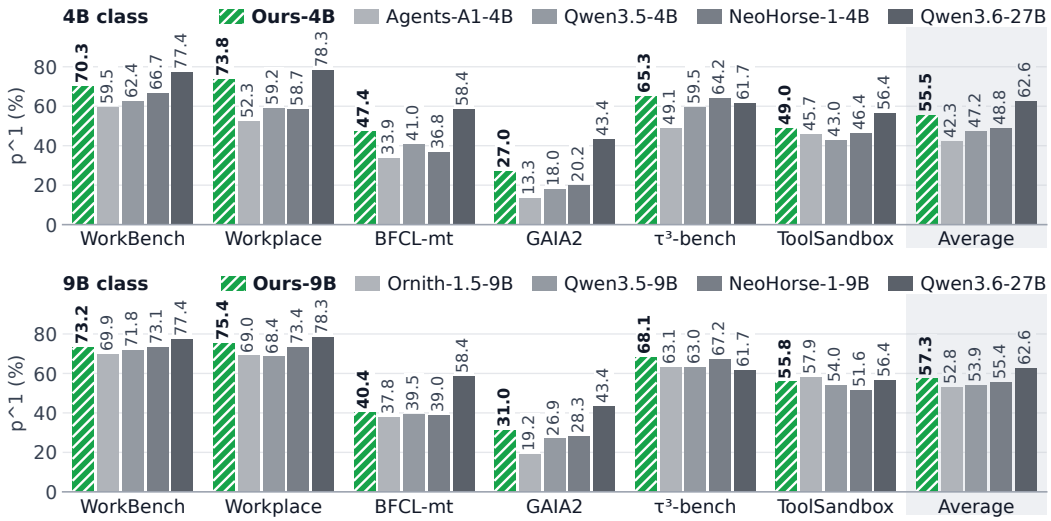



Figure 1: Success rate pass@k1 (%) on six tool-use benchmarks for our 4B and 9B models (green) and open models of similar size, with Qwen3.6-27B as a larger reference. Average: mean over the benchmarks measured for every averaged model in the row.

difficult. Chain length has limited effect, while requiring the model to make a complex decision, such as selecting a record or refusing, moves pass rates into the target band.

We train Qwen3.5 (Qwen Team, 2026) students from 0.8B to 9B on this corpus and evaluate them on six out-of-distribution benchmarks: WorkBench (Styles et al., 2024), Workplace Assistant (NVIDIA, 2025), GAIA2 (Froger et al., 2026), BFCL multi-turn (Patil et al., 2025), $\tau^3$-bench over airline, retail and telecom (Barres et al., 2025), and ToolSandbox (Lu et al., 2025) (Figure 1). Performance improves across all six benchmarks from 2B to 9B. The largest gains at 2B are on WorkBench (+20.1 points) and Workplace (+17.2), while 4B shows strong gains on Workplace (+14.6) and GAIA2 (+9.0), and 9B on Workplace (+7.0) and $\tau^3$-bench (+5.1). Our results show that a small corpus of roughly 200 sufficiently challenging tasks can yield substantial out-of-distribution gains at 4B and 9B through direct RL, without prior SFT. The same environments also support on-policy distillation, with a 4B student reaching approximately 98% of its 9B teacher's average performance across the evaluated benchmarks. The corpus also transfers beyond the Qwen family, with supervised fine-tuning substantially improving tool use in weaker Llama-3.2 (Meta AI, 2024) and SmolLM (Allal et al., 2025; Bakouch et al., 2025) models, whereas GRPO directly from these weak bases can inflate scores by learning to avoid tool use. The gains persist across different specification authors, code agents, and harnesses, while band-selected corpora consistently outperform random task sets of comparable size.

**Contributions.** (1) A verified RLVR factory: code agents write domains, tools, databases, and templates, and execution produces and replays every task instance, its gold trajectory, and its reward verdict, with no judge in the loop. (2) An evolution loop in which the same agents harden tasks into a learning band, with the finding that the need to make a complex decision, not chain length, makes a tool-use task hard. (3) Tool-use gains across Qwen3.5 models from 2B to 9B on all six out-of-distribution benchmarks, with particularly strong gains at 4B, and additional evaluations of three generation pipelines, knowledge retention, transfer to weaker models beyond the Qwen family, and knowledge transfer through on-policy distillation.

## 2 Related Work

**Agent Environments.** Modern agent research relies on realistic, domain-specific benchmarks that serve not only as evaluation standards but also as training environments. In the web domain, WebArena (Zhou et al., 2024), VisualWebArena (Koh et al., 2024), BrowserGym (de Chezelles et al., 2025), and WorkArena (Drouin et al., 2024) provide unified platforms for assessing an agent's

ability to complete multi-step tasks. In parallel, coding agent gyms such as SWE-bench (Jimenez et al., 2024) and SWE-Gym (Pan et al., 2025) draw tasks from real GitHub repositories and enable executable unit-test verification, making them suitable for reinforcement-learning training. In the human–computer interaction domain, OSWorld (Xie et al., 2024) and Windows Agent Arena (Bonatti et al., 2024) offer benchmarks for agents operating in real desktop operating systems. Finally, multi-domain suites such as AgentBench (Liu et al., 2024) and BFCL (Patil et al., 2025) evaluate web, code, and database interaction capabilities within a single framework.

Despite their realism, these environments are costly to build and maintain. This manual-engineering bottleneck has motivated the growing adoption of synthetic environment generation.

**Synthetic Environment Generation.** Recent work demonstrates that LLMs can synthesize entire training environments at scale. AutoEnv (Zhang et al., 2025) formulates environments as factorizable distributions over transitions, observations, and rewards, enabling efficient generation of heterogeneous worlds from a small set of primitives. In a complementary empirical line, RandomWorld (Sullivan et al., 2025) shows that agent performance scales predictably with the volume of procedurally generated tool-use data, providing evidence for the synthetic scaling hypothesis. Building on these foundations, Agent World Model (Wang et al., 2026b) introduces a fully synthetic pipeline that generates over 1,000 database-backed tool-use environments for everyday scenarios, while Agent-World (Dong et al., 2026) addresses the synthetic-to-real domain gap by aligning databases and tools with real-world task distributions.

Closing the loop between synthesis and verification, DIVE (Chen et al., 2026) inverts the usual order by executing real-world tools first and reverse-deriving tasks strictly entailed by the resulting traces, guaranteeing executability by construction. Its controlled scaling analysis further shows that diversity scaling outperforms quantity scaling for out-of-distribution generalization even with four times less data. AReaL-SEA (Gao et al., 2026) couples a similar verified-synthesis pipeline with RL training: a reflection loop refines synthesis and evaluation plans from failure attributions, each task ships with an executable checker that doubles as an RL reward, and fine-tuning the user simulator before GRPO proves critical for stability on $\tau^2$-bench. A complementary position paper of Liu et al. (2026) argues that such self-evolving loops sustain progress only when every iteration increases the *learnable information* available to the model, which requires asymmetric co-evolution of the proposer and verifier against the solver as well as proactive acquisition of external information.

Alternatively, WebDreamer (Gu et al., 2024) treats the LLM itself as the environment, using the agent's backbone to predict web navigation outcomes for lookahead planning. This avoids extra rollouts and cuts interaction cost, though prediction accuracy degrades over long horizons.

These synthetic environments serve dual roles: training agents via reinforcement learning and generating trajectories for supervised fine-tuning and distillation into compact models.

Recent baselines validate this approach. Agents-A1 (Team, 2026a) matches much larger models by scaling the planning horizon, while Nanbeige 4.2-3B (Lab et al., 2026) shows that a 3B model outperforms 9–12B peers after process-reward training on synthetic trajectories. Both depend on high-quality synthetic environments.

# 3 METHODOLOGY

## 3.1 VERIFIED ENVIRONMENT FACTORY

Figure 2 shows the loop. A builder agent writes an executable domain and a set of declarative task templates. A template names no customer, order, or row; it specifies a relation the generator resolves against a seeded world. The generator turns one template into a concrete request, derives the matching tool-call chain, and replays that chain before the task can join a training pool.

The builder agent writes the domain code, templates, and checks, but only while we design a domain or an evolution round. No language model runs when the generator instantiates a task, derives its gold, or scores a rollout. The gold policy scores $1.0$ on all tasks and an empty policy scores $0.0$.

**Gold construction** The generator derives every gold chain. It resolves the template's filter against a concrete world database of real rows and identifiers, emits the tool call the template mandates for

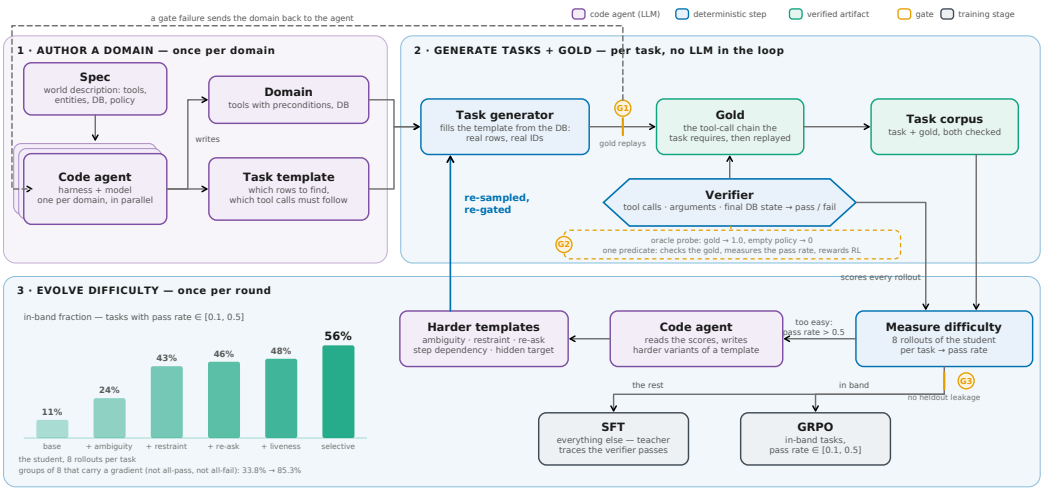



Figure 2: Verified task production and difficulty evolution. A builder agent writes domain code and declarative templates once per domain or evolution round. Deterministic programs then instantiate concrete tasks on seeded worlds, derive and replay gold trajectories, and score student rollouts. Measured pass rates select in-band tasks for GRPO and send saturated templates back for rule-level hardening; every evolved variant re-enters through the same replay gate

each matched row, and threads captured results through multi-step chains. It then replays every gold action against the live environment. The verifier scores a trajectory by matching gold action names and arguments in order, hashing the final database state, and flagging forbidden calls or extra writes, closing with a strict answer check.

**Factory outcome.** With Claude Code (Claude Opus 4.8 xhigh) as the builder, the factory produced 64 domain packages, 961 tools, and 6,989 tasks over nine template families. A second agent later rebuilt 19 of the same specifications independently (Section A.7). The tool code carries 2,037 explicit precondition checks, so when an agent tries an invalid write the environment refuses it and reports an error; the environment judges the action, and no language model reads the transcript. Every task takes one of the nine template shapes, and the mechanics that harden those shapes belong to Section 3.2.

### 3.2 Task evolution and difficulty

The last stage of Figure 2 keeps a verified task or hardens it. The current student runs eight rollouts on a candidate task, and the verifier turns them into a pass rate $p$. A task teaches GRPO only when $p$ sits in the band $0.1 \leq p \leq 0.5$: above it the eight rollouts succeed together and the group-relative advantage vanishes, below it they fail together. A task the student already solves does not become an RL example. The builder sees which check the task fails to exercise and adds a rule-level mechanic; the generator instantiates the harder variant, derives its gold, and replay-checks it from scratch. Tasks the student cannot yet reach still supply teacher trajectories the verifier accepts for SFT.

**Difficulty mechanics.** A mechanic rewrites a task's *specification* to push its pass rate into the band. The generator re-resolves the new specification against the world, derives fresh gold, and replay-checks it like any other task. We wrote five mechanics. Three produced corpus tasks, *join-depth* (2057 tasks), *ask-overlay* (908), and *hidden-target* (73); temporal windows and idempotence pass their tests but yielded no tasks we kept. Table 1 calibrates the two that carry the corpus: join-depth shows up as the *route dictated* and *route implied* rows, ask-overlay as the *decision to refuse* rows.

Figure 3 traces one such round. The student model solves the original request on every attempt, so the task carries no gradient. The evolved variant appends one sentence to escalate the acuity of "Daria Schroder" and two patients in the ward share that name, so the request cannot be carried out as stated. This type of mechanic touches no tool call: the gold chain stays byte-identical, while the verdict

Table 1: Each row is a separate corpus variant, scored by running the 4B student eight times on each of its $n$ tasks. *In band* is the fraction of tasks with $0.1 \leq p \leq 0.5$; *with gradient* is the fraction of eight-rollout groups that are neither all-pass nor all-fail. Deepening the tool-call chain while the request dictates the route leaves the band where it was; forcing the model to infer the route or to refuse a request raises it. The rows are separate corpora rather than successive edits of one, so the fractions are not cumulative. The full per-variant distribution and the rejected variants appear in Appendix A.

| corpus variant | $n$ | in band | with gradient |
|---|---|---|---|
| base corpus | 473 | 11.0% | 33.8% |
| deeper chain, route dictated | 683 | 6.4% | 29.9% |
| same depth, route inferred | 220 | 24.1% | 47.7% |
| one appended refusal | 219 | 43.4% | 66.7% |
| shipped corpus | 266 | 56.0% | 85.3% |

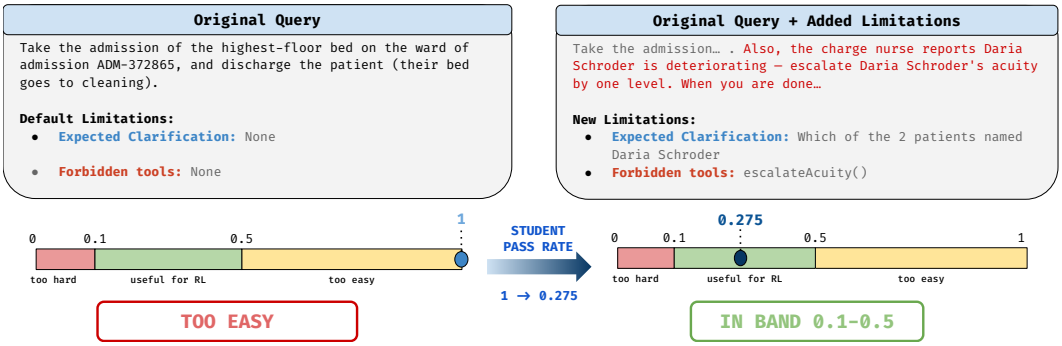



Figure 3: A hospital-ward task before and after one difficulty-evolution round. Gold tools call chain stays unchanged in this task, but learning signal is raising.

around it changes. Calling `escalateAcuity` now fails the task, and the expected behaviour is to ask which patient is meant rather than guess. The student's pass rate falls from 1.00 to 0.275 (11 of 40 pooled rollouts), into the band, while a 35B teacher still passes 0.875 (7 of 8): the task became hard, not broken.

Table 1 moves the in-band fraction from 11.0% to 56.0% and the fraction of eight-rollout groups carrying a gradient from 33.8% to 85.3%. One of the six shipped steps adds a mechanic; three tune a parameter of one, one is a gate, and one is a shipping rule, and those four together move the band less than the single mechanic. The difficulty tracks the decision a task forces, not its length. The shipping rule works because both screening extremes waste effort: an overlay does not rescue an unsolvable base (57.9% go dead) and does not harden a trivial one (15.1% stay saturated), while the contested middle lands in band 69.7% of the time. Appendix A.14 shows four more in-band tasks, built around a computed fee, an ambiguous reference, filter cardinality and a conditional branch (Figure 8).

**Hard is not broken.** A wrong gold, tool output, or user query would also show a low pass rate, so we separate difficulty from breakage. The request and the gold come from one resolution, and the generator replays the chain to a 1.0. As a second check we re-solve with the 35B teacher: of the 301 tasks measured on both models, 17 score zero for the 4B student, and the teacher passes 8 of them, so about half of what looks dead is hard rather than broken.

## 4 Experiments

**The reward.** The same predicate that verifies a task also scores a rollout during training, as strict pass or fail: the episode earns 1 when every applicable check holds at once and 0 otherwise. The checks are the gold write actions and their arguments, the final database state, the final answer, and

four guards: no forbidden tool called, every tool name real, at least one agent action, and no write outside the gold chain. The strict match falls only on the state-changing actions: read-only calls, searches and the re-checks an agent runs after a write may appear in any number or order without penalty. **Notation.** p^1 is the mean per-task score over a board's $k$ repeats, p^k the fraction of tasks correct in *all* $k$, and p@$k$ the fraction correct in at least one; each board is read at its native $k$.

**Training.** Our main models are the Qwen3.5 ladder at 0.8B, 2B, 4B, and 9B; we also run the same recipe on a weaker third architecture, Llama-3.2 3B (Meta AI, 2024) and SmolLM 2 and 3 (Allal et al., 2025; Bakouch et al., 2025), to check that it does not depend on one model family (Section 6). The weak Qwen tiers (0.8B, 2B) and the Llama base first warm up with SFT on 977 passing traces (446 tasks, 31 domains, one epoch); the 4B and 9B skip this step and train with GRPO directly on the in-band corpus. That corpus is 228 tasks over 45 domains: 198 training tasks over 37 domains and 30 validation tasks over a disjoint 8 domains, each inside the 4B-base learning band of Section 3.2. GRPO samples a group of eight rollouts per prompt, and the group's rewards set the advantage, so a prompt moves the weights only when its rollouts disagree; outside the band the group agrees and the advantage vanishes. Each model fits a LoRA adapter with rank 32; Table 14 in the appendix gives every optimizer setting.

**Held out.** Inside training, 30 tasks over 8 domains are withheld from every arm for validation. Across evaluation, no domain or tool the models train on appears on any external board, apart from a generic delivery tools, so every number in Section 5 measures transfer to unseen domains.

Our main evaluation covers five tool-use settings: **Gaia2**, **WorkBench**, **Workplace Assistant**, **BFCL Multi-Turn**, and $\tau^3$**-bench** on Airline, Retail and Telecom. We additionally evaluate **ToolSandbox**. Appendix Table 5 summarizes these settings.

**Gaia2** (Froger et al., 2026) is our main high-complexity benchmark. The agent receives all 101 tools in context and solves long multi-step tasks in a dynamic environment. *Search* tasks require gathering information across multiple actions, while *Ambiguity* tasks involve underspecified or conflicting requests; other scenarios introduce state changes and temporal constraints. Even frontier models perform poorly: the strongest model in the original evaluation reaches only about 42% pass@1. Our results use a fixed harness and are not directly comparable to this figure or the public leaderboard; Section A.2 details the differences.

**WorkBench** (Styles et al., 2024) and **Workplace Assistant** (NVIDIA, 2025) provide practice-oriented evaluation. WorkBench contains 690 tasks over five workplace databases and 26 tools covering email, calendar, CRM, project management, and analytics, with success determined by the final environment state. Workplace Assistant is NVIDIA's executable NeMo Gym adaptation of the same setting. We include both to test similar business workflows across different benchmark implementations and harnesses. Our WorkBench setup uses domain-scoped tools and non-greedy multi-sample decoding, so its results are not directly comparable to the published leaderboard.

For **BFCL** (Patil et al., 2025), we use only the **Multi-Turn** evaluation from v4 release. It spans diverse API domains such as file operations, travel, vehicle control, trading, and messaging, with *Base*, *Missing Parameter*, *Missing Function*, and *Long Context* settings. These isolate state retention, missing information, unavailable tools, and tool selection under long context. BFCL also includes predefined multi-turn interactions with persona-based wording, providing a lightweight simulation of continued user interaction.

$\tau^3$**-bench** (Barres et al., 2025) adds an LLM-based user simulator. We evaluate its *Airline*, *Retail* and *Telecom* domains, where agents handle bookings, orders, returns, and exchanges while interacting with users, calling domain-specific tools, and following explicit business policies.

**ToolSandbox** (Lu et al., 2025) combines a simulated user with persistent environment state and dependencies between tools. Evaluation checks intermediate and final milestones and forbidden actions. Appendix A.4 describes our scenario subset and harness modifications.

Overall, the suite covers long-horizon interaction with large tool spaces, realistic workplace automation, controlled multi-turn failure modes, and conversational tool use with simulated users.

**Capability retention.** We compare the base Qwen3.5-4B checkpoint with ours on five general benchmarks and IFBench (Pyatkin et al., 2025), which tests precise instruction following using programmatically verifiable response constraints. Appendix A.6 reports the setup and results.

Table 2: Main results (%). Each block lists a Qwen3.5 base model, our model at that size (Ours) and open models of similar size; Qwen3.6-27B is a larger reference. $k = 3$ for WorkBench and GAIA2, $k = 5$ for Workplace and $\tau^3$; Bold: best value within a block. Non-thinking inference regime.

| | WorkBench | | | Workplace | | | BFCL | GAIA2 | | | $\tau^3$ | | | Tool Sandbox |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | p^1 | p^3 | p@3 | p^1 | p^5 | p@5 | p^1 | p^1 | p^3 | p@3 | p^1 | p^5 | p@5 | p^1 |
| Qwen3.5-2B | 22.9 | 11.9 | 35.7 | 29.4 | 8.6 | 54.9 | 17.5 | 4.1 | **0.9** | 8.6 | 31.9 | 8.6 | 66.2 | 36.0 |
| Ours-2B | **43.0** | **25.4** | **59.9** | **46.6** | **22.2** | **71.7** | **25.3** | **4.5** | **0.9** | **9.5** | **37.8** | **13.0** | **72.1** | **36.3** |
| Qwen3.5-4B | 62.4 | 48.3 | 76.4 | 59.2 | 37.2 | 79.1 | 41.0 | 18.0 | 6.3 | 31.5 | 59.5 | 30.8 | 84.5 | 43.0 |
| Ours-4B | **70.3** | **57.2** | **82.3** | **73.8** | **53.2** | **89.5** | **47.4** | **27.0** | **13.1** | **42.0** | **65.3** | **38.5** | 87.5 | **49.0** |
| Agents-A1-4B | 59.5 | 45.2 | 73.6 | 52.3 | 29.7 | 73.8 | 33.9 | 13.3 | 4.8 | 23.4 | 49.1 | 20.4 | 80.9 | 45.7 |
| NeoHorse-1-4B | 66.7 | 53.2 | 78.4 | 58.7 | 33.2 | 82.4 | 36.8 | 20.2 | 7.5 | 35.6 | 64.2 | 32.0 | **88.3** | 46.4 |
| Qwen3.5-9B | 71.8 | 60.4 | 81.7 | 68.4 | 47.7 | 86.2 | 39.5 | 26.9 | 13.8 | 42.3 | 63.0 | 36.7 | 85.3 | 54.0 |
| Ours-9B | **73.2** | **63.6** | **82.5** | **75.4** | **57.8** | 87.2 | **40.4** | **31.0** | **17.8** | **45.8** | **68.1** | **43.4** | 86.6 | 55.8 |
| Ornith-1.5-9B | 69.9 | 57.8 | 81.6 | 69.0 | 49.9 | 83.7 | 37.8 | 19.2 | 6.9 | 34.0 | 63.1 | 30.0 | **92.1** | **57.9** |
| NeoHorse-1-9B | 73.1 | 62.3 | 81.9 | 73.4 | 55.0 | **88.4** | 39.0 | 28.3 | 14.1 | 44.5 | 67.2 | 37.0 | 90.8 | 51.6 |
| Qwen3.6-27B | 77.4 | 70.1 | 85.1 | 78.3 | 66.8 | 87.9 | 58.4 | 43.4 | 28.3 | 58.3 | 61.7 | 30.9 | 86.4 | 56.4 |

# 5 RESULTS

## 5.1 MAIN RESULTS

Table 2 groups the models by size. Each block lists the Qwen3.5 base, our main model at that size (+ Ours), our other training runs from Sections A.9 and 5.2, and open models of similar size; Qwen3.6-27B and Qwen3.6-35B-A3B serve as larger references. At 2B, OPD from a 4B teacher raises WorkBench from 22.9 to 49.5 and Workplace from 29.4 to 50.4, and it scores above our SFT + GRPO run on WorkBench, Workplace, BFCL and ToolSandbox. At 4B, + Ours improves on the base on all six boards, with GAIA2 rising from 18.0 to 27.0. It also scores above every open 3–4B model on WorkBench, Workplace, BFCL and GAIA2; the strongest of them, NeoHorse-1-4B, reaches 66.7, 58.7, 36.8 and 20.2 on these boards against our 70.3, 73.8, 47.4 and 27.0. NeoHorse-1-4B comes within 1.1 points of + Ours on $\tau^3$ (64.2 against 65.3), and Nanbeige-4.2-3B has the highest ToolSandbox score in the block (57.1 against 49.0). Our two alternative 4B runs score higher than + Ours on BFCL (51.3 with the GLM-specified corpus, 50.9 with OPD), while + Ours keeps the highest GAIA2 and $\tau^3$ scores among our runs. At 9B, + Ours improves on the base on WorkBench, Workplace, BFCL, GAIA2 and ToolSandbox, with the largest gains on Workplace (68.4 to 75.4) and GAIA2 (26.9 to 31.0). NeoHorse-1-9B nearly matches it on WorkBench (73.1 against 73.2), trails it on Workplace, BFCL and ToolSandbox, and has the highest $\tau^3$ score in the block (67.2); Gemma-4-12B has the highest Workplace score (78.1). GRPO from the 9B base on the GLM-specified corpus scores higher than + Ours on WorkBench (75.3) and BFCL (45.9) and lower on GAIA2 (26.9). The larger Qwen3.6-27B has the highest WorkBench, Workplace, BFCL and GAIA2 scores in the table. A paired task-level bootstrap against the base (Appendix A.12) separates every 4B gain from zero, and the WorkBench, Workplace and BFCL gains at 2B. At 9B only the Workplace and GAIA2 pass@k1 gains exclude zero.

**Training recipes.** Appendix A.9 compares cold GRPO, SFT + GRPO and OPD at 0.8B, 2B and 4B (Table 12). Small bases need a warm start: at 0.8B cold GRPO lowers WorkBench from 15.0 to 10.0, while SFT followed by GRPO raises it to 27.0. At 4B the base already calls tools, and GRPO from the base gives the highest GAIA2 and $\tau^3$ scores among the recipes (27.0 and 65.3), so our main 4B model uses it.

## 5.2 SPECIFICATION BUILDER, CODE AGENT AND HARNESS

Table 3 compares three generation pipelines, each training the same 4B base with GRPO. In the reference pipeline, Claude Fable 5 writes the specifications and Claude Opus 4.8 the code inside

Table 3: Generation pipelines and downstream 4B GRPO performance (p^1, %; WB: WorkBench, WP: Workplace). Dom. and Tasks count the domains and tasks of the training corpus. Bold: best value per column; dashes: not evaluated.

| Spec builder | Code agent | Harness | Dom. | Tasks | WB | WP | BFCL | GAIA2 | $\tau^3$ |
|---|---|---|---|---|---|---|---|---|---|
| Claude Fable 5 | Claude Opus 4.8 | Claude Code | 17 | 129 | 70.3 | 73.8 | 47.4 | **27.0** | **65.3** |
| Claude Fable 5 | Claude Opus 4.8 | Claude Code | 40 | 217 | **70.9** | 71.4 | **51.5** | 25.1 | – |
| GLM 5.3 | DeepSeek V4 Pro | DeepSeek Harness | 32 | 158 | **70.9** | **74.2** | 51.3 | 23.3 | 60.8 |

Table 4: In-band versus random task selection for the 4B model (p^1, %, mean $\pm$ sample standard deviation over three GRPO runs). In-band: our three in-band corpora of 129, 158 and 217 tasks; Random: three random draws of 198 tasks each.

| Selection | WorkBench | Workplace | BFCL |
|---|---|---|---|
| In-band | $70.7 \pm 0.3$ | $73.1 \pm 1.5$ | $50.0 \pm 2.3$ |
| Random | $66.4 \pm 0.7$ | $65.9 \pm 0.6$ | $40.2 \pm 6.3$ |

Claude Code; we train on three corpora from it, with 49 tasks over 12 domains, 129 over 17 (our main 4B model) and 217 over 40. In the second, DeepSeek V4 Pro writes both the specifications and the code inside DeepSeek Harness (DSH), an open-source headless coding agent, starting from seeded domain assignments, with no Claude specification or code in its prompt. In the third, GLM 5.3 writes the specifications and DeepSeek V4 Pro implements them in DSH. Every corpus trains the 4B model above its base on WorkBench, Workplace and BFCL, and on GAIA2 and $\tau^3$ wherever we measured them, but the pipelines lead on different boards. The 129-task Claude corpus gives the highest GAIA2 and $\tau^3$ scores (27.0 and 65.3, against 23.3 and 60.8 for the GLM corpus), and our main 4B model uses it for that reason. Specifications from GLM 5.3 implemented by DeepSeek V4 Pro give the highest Workplace score (74.2) and nearly the highest BFCL score (51.3 against 51.5), which makes this pipeline a strong alternative builder. Within the Claude pipeline, the 217-task corpus scores above the 49-task one on Workplace (71.4 against 68.0) and BFCL (51.5 against 48.1), while WorkBench stays within 0.6 points across all three corpora. The 129-task corpus does not fall between them on these two boards (73.8 and 47.4), so task count and domain breadth do not explain the differences alone. Corpus sizes and training budgets differ, so the table compares complete pipelines rather than isolating the specification author or code agent. Appendix A.7 adds a DSH rebuild of 20 Claude specifications, which cleared the build gates on 19 domains with no manual intervention for $60.77 in total.

### 5.3 In-band versus random selection

Table 4 trains the 4B model on band-selected and on randomly drawn tasks. The in-band row averages our three in-band corpora, two built with Claude and one specified by GLM 5.3 (129 to 217 tasks), and the random row averages three random draws of 198 tasks each, trained for 30 steps against 30 to 50 for the in-band runs. In-band selection has the higher mean on every board. On BFCL the random draws also vary far more: their standard deviation is 6.3 points and their scores span 32.9 to 44.5, while the in-band runs keep a standard deviation of 2.3 although their corpora differ in builder and size. The spread across runs measures how much training varies; the task-level bootstrap intervals in Appendix A.12 measure a different variance, the sampling of tasks behind one fixed checkpoint.

## 6 Transfer to non-Qwen bases

The recipe carries over to bases outside the Qwen family. Llama-3.2-3B and SmolLM3-3B start below every Qwen tier on WorkBench (7.6 and 11.1 pass@k1). Supervised fine-tuning on the corpus traces lifts them to 21.8 and 32.8 (Figure 4), and Workplace rises for all three non-Qwen bases. We also tried GRPO directly from these bases, without SFT. It does not install tool use: Llama-3.2-3B climbs from 7.6 to 18.0 p^1 on WorkBench, exactly the score of a policy that never calls a tool, 73% of the SmolLM3-3B score comes from rollouts without a tool call, and at 0.8B the same GRPO lowers WorkBench from 15.0 to 10.0 (Table 12). After SFT, GRPO keeps the gains earned (Appendix A.11).

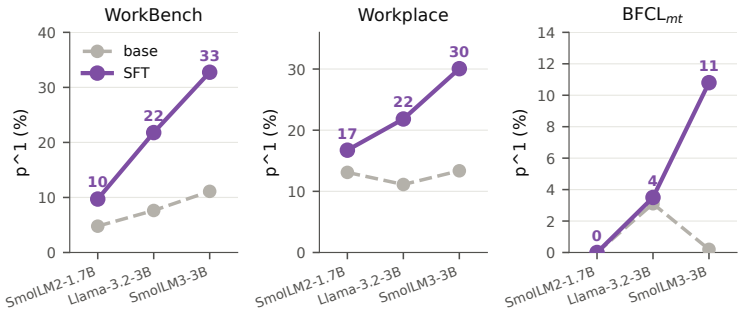



Figure 4: p^1 on WorkBench, Workplace and BFCL multi-turn of three weak non-Qwen bases before (gray, dashed) and after SFT on the corpus traces (purple, solid).

## 7 Limitations

This study leaves several things untested. The largest student we train is 9B, so whether the gains persist at frontier scale is open. The corpus is modest, 129 in-band RL tasks over 17 domains, and the evolution loop ran once at this size rather than at the hundreds of domains more compute would reach. The evolution loop ran only with Claude as the coding agent; the DeepSeek builder implemented domains from ready-made specs rather than evolving them, so the cross-builder evidence is about domain generation, not evolution. We report the build cost only for that DeepSeek run ($60.77 over 20 domains) and cannot report the cost of the Claude Code build, so the builder comparison in Table 3 speaks to corpus quality, not cost-efficiency. Every student is a rank-32 LoRA adapter rather than a full-parameter finetune. We train and evaluate every model with reasoning disabled, although the factory and its deterministic reward place no such restriction and thinking-mode behaviour is unmeasured. Finally, we validate a single policy-optimization method, GRPO; the corpus and its reward are agnostic to the optimizer, but alternatives such as GSPO remain untested. We will release the generated domains and their task files and the evaluation harnesses.

## 8 Conclusion

We turn code agents into a source of verified RLVR data. The agents author a domain, its tools, a seeded database, and task templates, and deterministic execution instantiates every task, derives its gold chain, and scores it through one predicate that also measures difficulty and pays the reward, with no judge in the loop. The measured pass rate serves two purposes: only tasks inside the learning band enter the training corpus, and the saturated ones are sent back to the code agents for hardening. Trained on the band-calibrated corpus Qwen3.5 improves on the three execution-heavy boards at every size and on all six from 4B up; the recipe also transfers to weak non-Qwen bases once SFT installs the tool-calling that RL alone cannot. The corpus is small and the loop ran once. Scaling the evolution to hundreds of domains is future work, and the factory makes that a matter of compute rather than annotation.

### AI use statement

Code agents are part of the method. Claude Code with Claude Fable 5 and Claude Opus 4.8, DeepSeek Harness with DeepSeek V4 Pro, and GLM 5.3 write the domain specifications, tool code and task templates that the factory then verifies (Sections 3.1 and 5.2). Deterministic code instantiates the tasks, derives their gold chains and computes every reward; no language model judges training rollouts. Language models also act as students, teachers, user simulators and benchmark judges, as Section 4 and Appendix A.2 describe. Beyond the method, we used LLM assistants to write and debug experiment code, to aggregate and check evaluation results, and to edit the text of this paper. The authors reviewed all AI-assisted code, analyses and text, and take responsibility for the final content of this work.

### Reproducibility statement

Section 3.1 and Appendix A describe the factory, the verifier and the evolution loop. Section 4 and Appendices A.2 and A.3 give the evaluation harnesses, our modifications to them and the task coverage of every reported run. Appendix 14 lists the training configuration, and Appendix A.7 documents the DeepSeek-Harness build campaign with its gates and costs. We will release the generated domains, their task files, the verifier and the evaluation harnesses.

## References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. $\tau^2$-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024.

Aili Chen, Chi Zhang, Junteng Liu, Jiangjie Chen, Chengyu Du, Yunji Li, Ming Zhong, Qin Wang, Zhengmao Zhu, Jiayuan Song, et al. Dive: Scaling diversity in agentic task synthesis for generalizable tool use. *arXiv preprint arXiv:2603.11076*, 2026.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *CoRR*, abs/2604.18292, 2026. doi: 10.48550/ARXIV.2604.18292. URL https://doi.org/10.48550/arXiv.2604.18292.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking llm agents on dynamic and asynchronous environments, 2026. URL https://arxiv.org/abs/2602.11964.

Jiaxuan Gao, Jiaao Chen, Chuyi He, Shusheng Xu, Di Jin, and Yi Wu. From self-evolving synthetic data to verifiable-reward rl: Post-training multi-turn interactive tool-using agents. *arXiv preprint arXiv:2601.22607*, 2026.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*, 2024.

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

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Nanbeige Lab, Chen Yang, Chengrui Huang, Fufeng Lan, Hanhui Chen, Hao Zhou, Huatong Song, Jiaqi Cao, Jiaying Zhu, Jinlin Niu, et al. Nanbeige4. 2-3b: Unlocking agentic capabilities in a compact model. *arXiv preprint arXiv:2607.22083*, 2026.

Wei Liu, Siya Qi, Yali Du, and Yulan He. Self-play only evolves when self-synthetic pipeline ensures learnable information gain. *arXiv preprint arXiv:2603.02218*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Anas Mahmoud, MohammadHossein Rezaei, Zihao Wang, Anisha Gunjal, Bing Liu, and Yunzhong He. Reward hacking in rubric-based reinforcement learning. In *Second Workshop on Agents in the Wild: Safety, Security, and Beyond*, 2026. URL https://openreview.net/forum?id=X5sPkeE60f.

Meta AI. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

NVIDIA. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-gym. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Cq1BNvHx74.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/5e3661f7fe4c8ac5652d62eb3d3c96ea-Abstract-Datasets_and_Benchmarks_Track.html.

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

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. Workbench: a benchmark dataset for agents in a realistic workplace setting. In *Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=4HNAwZFDcH.

Michael Sullivan, Mareike Hartmann, and Alexander Koller. Procedural environment generation for tool-use agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 18544–18562. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.936. URL https://doi.org/10.18653/v1/2025.emnlp-main.936.

Agents-A1 Team. Scaling the horizon, not the parameters: Reaching trillion-parameter performance with a 35b agent. *CoRR*, abs/2606.30616, 2026a. doi: 10.48550/ARXIV.2606.30616. URL https://doi.org/10.48550/arXiv.2606.30616.

Kimi Team. Kimi K3: open frontier intelligence. *CoRR*, abs/2607.24653, 2026b. doi: 10.48550/ARXIV.2607.24653. URL https://doi.org/10.48550/arXiv.2607.24653.

NeoHorse Team, Guoliang Cao, Guohao Dai, Tianyu Guo, Kai Han, Hailin Hu, Zihan Jiang, Xiang Kuang, Boxun Li, Yulong Li, et al. Neohorse-1: Towards recursive self-improvement via agentic post-training with routing harness. *arXiv preprint arXiv:2609.08183*, 2026.

Xuekang Wang, Zhuoyuan Hao, Shuo Hou, Hao Peng, Juanzi Li, and Xiaozhi Wang. Reproducing, analyzing, and detecting reward hacking in rubric-based reinforcement learning. *CoRR*, abs/2606.04923, 2026a. doi: 10.48550/ARXIV.2606.04923. URL https://doi.org/10.48550/arXiv.2606.04923.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *CoRR*, abs/2602.10090, 2026b. doi: 10.48550/ARXIV.2602.10090. URL https://doi.org/10.48550/arXiv.2602.10090.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Xingshan Zeng, Weiwen Liu, Lingzhi Wang, Liangyou Li, Fei Mi, Yasheng Wang, Lifeng Shang, Xin Jiang, and Qun Liu. Toolace-mt: Non-autoregressive generation for agentic multi-turn interaction. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/065cb28127caf25d9f186f5460d7bb9c-Abstract-Conference.html.

Jiayi Zhang, Yiran Peng, Fanqi Kong, Cheng Yang, Yifan Wu, Zhaoyang Yu, Jinyu Xiang, Jianhao Ruan, Jinlin Wang, Maojia Song, HongZhang Liu, Xiangru Tang, Bang Liu, Chenglin Wu, and Yuyu Luo. Autoenv: Automated environments for measuring cross-environment agent learning. *CoRR*, abs/2511.19304, 2025. doi: 10.48550/ARXIV.2511.19304. URL https://doi.org/10.48550/arXiv.2511.19304.

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

# A Appendix

## A.1 The evaluation suite

The six boards, with the tool surface each exposes and the trajectory statistics our harness measures on it.

Table 5: Summary of the evaluated benchmark settings. Context and trajectory statistics are measured under our evaluation harness.

| Benchmark | Tasks | Tools | Avg. turns | Avg. tool calls | Avg. context |
|---|---|---|---|---|---|
| Gaia2 | 800 | 101 | 27 | 27 | 42k |
| WorkBench | 690 | 26 | 5 | 4 | 4k |
| Workplace Assistant | 545 | 27 | 4 | 3 | 21k |
| BFCL Multi-Turn | 800 | 128 | 13 | 9 | 10k |
| $\tau^3$ | 278 | 30 | 16 | 8 | 12k |
| ToolSandbox | 509 | 27 | 7 | 5 | 2k |

## A.2 GAIA2 harness fixes

We serve GAIA2 through our own harness, so our absolute numbers are not comparable to the public leaderboard. Table 6 takes that harness apart in three stages. Two of the three changes are bug fixes rather than tuning: the agent's own tool calls are replayed to it as native tool-call messages instead of as text, and every call a turn emits is executed rather than only the first, where the earlier harness ran the first and dropped the rest.

The two fixes pay off at opposite ends of the model range, which is itself informative. A small model gains almost everything from the protocol change: under the text protocol it few-shot-imitates the malformed JSON sitting in its own history, narrates "done" instead of calling the finish tool, and loops to the turn cap, which the 2B did on $95\%$ of scenarios before the change and $43\%$ after. A large model gains from parallel execution instead, in proportion to how far ahead it plans, so that penalty grows with size. Either effect is large enough that a GAIA2 number quoted without its harness is not interpretable, which is why we state ours and compare only against our own base.

Table 6: The agent harness, taken apart in three stages, and what each stage did to the GAIA2 score. Custom self-hosted harness; validation $= 5$ capabilities $\times 160$ scenarios, macro p^1, judge Llama-3.3-70B, thinking off. **ReAct** puts the tools in a text system prompt and replays past steps as text; **Native** passes real tool schemas through the chat template and replays the model's own tool-call and tool-response messages; +**Parallel** executes every call a turn emits instead of the first alone. The values come from the harness-development runs; Table 2 uses a later run of the final harness, which gives the 2B and 4B bases 4.1 and 18.0. Section A.2 reads the two effects.

| Base model | GAIA2 macro p^1 (%) | | |
|---|---|---|---|
| | ReAct | Native | + Parallel |
| Qwen3.5-2B | 0.4 | 3.2 | **3.7** |
| Qwen3.5-4B | 14.3 | 16.8 | **17.6** |
| Qwen3.5-9B | 22.0 | 24.0 | **26.9** |
| Qwen3.6-35B-A3B | – | 26.5 | **35.7** |

### A.3 $\tau^3$ AIRLINE, RETAIL, AND TELECOM EVALUATION

We evaluate Airline (50 tasks), Retail (114 tasks), and Telecom (114 tasks from the base split), using the original $\tau^3$-bench task definitions (Barres et al., 2025). Qwen3.6-35B-A3B-FP8 serves as both the user simulator and the natural-language judge, with greedy decoding and thinking disabled. The evaluated agent also uses greedy decoding ($T = 0$), with a 4096-token limit per generation. We run five trials per task, allowing up to 120 steps for Airline and Telecom and 80 for Retail.

For Airline and Retail, we add a `send_message` tool to handle user-facing replies emitted as tool calls. It forwards the message to the user without changing the environment state, avoiding failures caused by the response format. Telecom uses no such wrapper and retains the original $\tau^3$ setup, in which the user simulator has its own domain tools.

### A.4 TOOLSANDBOX EVALUATION

We evaluate 509 scenarios from the `local` subset, which excludes scenarios requiring RapidAPI. The user simulator is `ProxyUser` with the `qwen` preset, backed by Qwen3.6-35B-A3B-FP8 with thinking disabled. The agent runs in vLLM 0.24.0 with the `qwen3_xml` parser, thinking disabled, and temperature 0. We collect five trajectories per scenario with seeds 42–46 and retain the default limit of 30 messages.

All runs enable `-bench-fixes`. Unknown tools and malformed arguments return an error to the agent; ten protocol errors end the conversation. Holiday names are treated as equivalent when they refer to the same date. Insufficient-information scenarios accept a refusal without requiring the reference wording. Agent messages to the user are checked for factual agreement using factual checks and an LLM judge, replacing ROUGE-based reference matching. These changes make our scores specific to this harness and not directly comparable to the original ToolSandbox results.

### A.5 CORPUS VARIANTS MEASURED DURING EVOLUTION

Every variant we measured on the way to the shipped corpus, including the ones we rejected, with the pass-rate distribution each produced.

Table 7: How the corpus was moved into the learning band. Each row is a whole corpus variant, measured by running the 4B student eight times on each of its $n$ tasks, except the 35B row, which is a teacher on a different protocol and not comparable in absolute terms. Grey rows we measured and did not ship. *Varies* says what each row changed: of the six that shipped, one is the unmodified base corpus and exactly one of the five changes adds a mechanic. Rows are not cumulative, $n$ differs, and the intervals use $z = 2$. The last three rows split the shipped variant by a four-rollout screen run before the mechanic was applied, which is the evidence for shipping only the contested middle; the middle row aggregates 1, 2 and 3 of 4 weighted by $n$.

| | | | % of tasks whose pass rate $p$ is | | | groups with |
|---|---|---|---|---|---|---|
| arm | varies | $n$ | 0 *dead* | $0.1-0.5$ **in band** | $> 0.9$ *saturated* | a gradient (%) |
| base corpus | – | 473 | 11.6 | 11.0 | 54.5 | 33.8 |
| join depth, route dictated | param | 683 | 0.9 | 6.4 | 69.3 | 29.9 |
| join depth, route implied | param | 220 | 11.8 | 24.1 | 40.5 | 47.7 |
| + one decision to refuse | **mechanic** | 219 | 22.8 | **43.4** | 10.5 | 66.7 |
| + two decisions to refuse | param | 220 | 22.7 | 35.0 | 20.0 | 57.3 |
| + answer not re-asked | param | 672 | 22.9 | 45.4 | 9.5 | 67.6 |
| + answer re-asked | param | 671 | 20.9 | $46.1 \pm 3.8$ | 8.6 | 70.5 |
| + liveness gate | gate | 564 | 13.8 | $47.9 \pm 4.2$ | 10.1 | 76.1 |
| + ship contested only | ship rule | 266 | 10.2 | $56.0 \pm 6.1$ | 4.5 | 85.3 |
| the shipped set, read on the 35B teacher | readout | 113 | 19.5 | $45.1 \pm 9.4$ | 4.4 | 76.1 |
| screen 0 of 4, unsolvable | – | 107 | 57.9 | 36.4 | 0.9 | – |
| screen 1–3 of 4, shipped | – | 249 | 10.8 | **55.0** | 4.4 | – |
| screen 4 of 4, trivial | – | 298 | 17.1 | 40.6 | 15.1 | – |

### A.6 Capability retention

We compare Qwen3.5-4B before and after GRPO. The five general benchmarks use the original Open LLM Leaderboard protocol: raw completions without a chat template. IFBench separately tests instruction following through programmatic checks of response constraints.

Table 8: Capability retention. All scores are percentages.

| Benchmark | Base | Ours |
|---|---|---|
| MMLU | 70.2 | 69.7 |
| HellaSwag | 74.8 | 74.8 |
| ARC | 65.1 | 65.2 |
| WinoGrande | 69.2 | 69.0 |
| TruthfulQA | 48.8 | 48.6 |
| IFBench | 32.9 | 33.5 |

### A.7 The DeepSeek-Harness build campaign

Figure 5 reports an earlier, separate ablation: the DeepSeek Harness agent rebuilding twenty ready-made Claude specifications. The campaign below removes the remaining dependence on Claude and generates its domains from scratch. GLM agent distilled the sixty-five Claude-built domains into a catalogue of the rules, a verification ladder that ends in a quota, multi-level approval, dependency graphs, date windows checked across records, eligibility that recomputes along links, role quorums, settlement arithmetic, and bookable capacity. afterwords, thirty-three everyday back-office thematic setups were generated. A seeded sampler then deals each new domain a pairing of structure and setting that no earlier domain used, with budgets and a train or held-out stamp. DeepSeek V4 Pro, through the same agent, expands the assignment into a specification and code, and deterministic gates verify the expansion against it: gold chains must replay, a 27B teacher must still solve the tasks, and the student's pass rates determine tasks for RL. Four waves consumed 107 assignments and yielded 39 calibrated domains with 1,099 tasks, eight rollouts each. The difficulty ladder below was cheap

to climb: every control is a field of the deterministic assignment, so specification and build stayed at a median of $2.27 per domain, and the full cycle from assignment to calibration fell from about thirteen hours to five and a half once scoring ran as parallel.

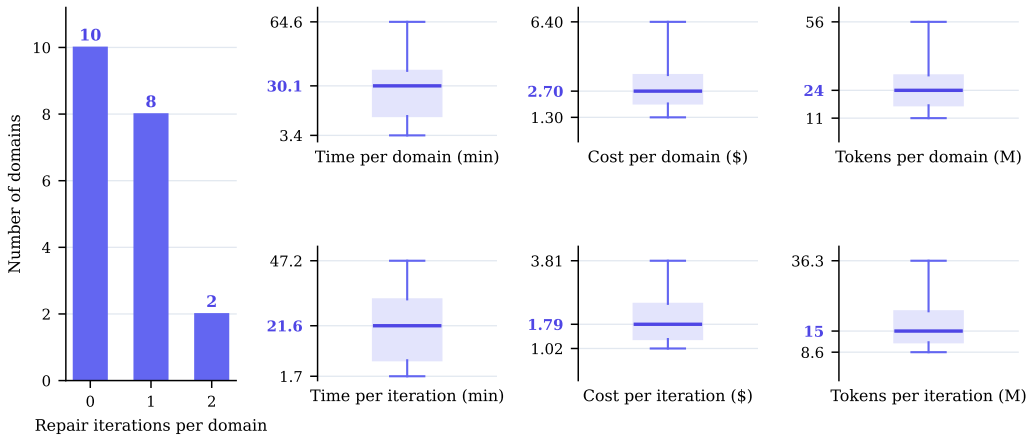



Figure 5: The DeepSeek-Harness generation campaign over 20 domains: gate outcomes, repair iterations, build cost, and tokens per domain. Nineteen clear the gates G1–G3, ten with no repair, at $60.77 and 530M tokens with no manual intervention.

### A.8 Pass rates on the rebuilt corpus

On the 560 tasks of that specification rebuild (Table 9) the teacher solves 79% at least once and the student 74%; 72 tasks (13%) sit in the learning band. In the self-generated releases, difficulty became a controlled variable (Table 10): the controls evolved in three versions (Table 11).

The base controls of the first version were solution chains of four to seven tool calls, filtered row sets of fifteen to twenty-five records, one trap, one type of twin records, and monetary fields in about 80% of domains. A trap is a step that looks legal but is refused because a condition quietly fails — the quota is already spent or the deadline has passed. Twin records are near-duplicate entries that make the name in a request match two rows at once, and the correct move is to ask which one is meant rather than guess. These controls left the student saturated: mean pass 0.89, in-band empty. The second version added a second trap placed elsewhere in the chain, a second twin type, monetary fields in every domain, and larger row sets with bulk collections; the in-band share rose to 13.1% while the median number of write actions stayed put, so the difficulty came from new decisions rather than length. The third version added the two controls that load computation. A fee formula makes the payment a sum of two or three components that the tool re-derives from the record and rejects on any mismatch. A multi-hop reference hides the identifier a write needs behind two or three chained searches. The third version ran in two waves — the third and fourth releases, with identical controls. As a result the in-band share raised to 16.9% and 14.9%, the share solved at least once fell from 93% to 73%, and the share solved every time from 70% to 33%.

Within a release, the band follows the kind of task, not the amount of arithmetic. Two families fill two thirds of the in-band pool: tasks that ask the agent to disambiguate, and chains whose first write creates an identifier that every later step needs. Monetary tasks are no harder than average, and the one control that lowers the pass rate of both student and teacher, a row set of fifteen records or more, covers only 2.4% of tasks. A new rung was accepted only when the teacher could still solve the tasks it produced and the in-band share grew. The check added no extra model work, so difficulty rose while the cost per domain remained unchanged.

| | Tasks | Teacher 27B | | | | | | Student 4B | | | | | | Band |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | pass@, % | | | pass^, % | | dead, % | pass@, % | | | pass^, % | | dead, % | $[0.1, 0.5]$ |
| | | 1 | 4 | 8 | 4 | 8 | | 1 | 4 | 8 | 4 | 8 | | tasks |
| **Total (19 of 20 built)** | **560** | **68** | **77** | **79** | **56** | **50** | **21** | **61** | **72** | **74** | **47** | **40** | **25** | **72/560** |

Table 9: Cumulative evaluation of the 19 validated domains out of the 20 built (560 tasks). Both models ran eight times per task; the student is scored by the deterministic trainer reward. All values are percentages: pass@$k$ is the expected share of tasks solved in at least one of $k$ runs (unbiased estimator over the eight runs per task), pass^$k$ the share solved by all $k$ runs, and dead the share of never-solved tasks (the teacher screened 555 of the 560 tasks). The band counts tasks with a student pass rate in $[0.1, 0.5]$.

Table 11: The difficulty controls that every generated domain receives, shown for each of the three versions of the assignment generator. Each control is a field of the deterministic assignment, so a new version changes what the builder is asked for, not how much building costs. The controls of a version apply to every domain of its waves.

| Control | v1 | v2 | v3 |
|---|---|---|---|
| Chain length (calls on the shortest correct path) | 4–7 | 6–9 | 6–9 |
| Large filtered row set | 15–25 | 18–40 | 18–40 |
| Bulk collection inside one row set | $\geq$25 rows | $\geq$35 rows | $\geq$35 rows |
| Rows per entity type | 8–45 | 8–60 | 8–60 |
| Writing tools | 5–9 | 5–10 | 5–10 |
| Traps (legal-looking step refused) | 1 | 2 | 2 |
| Twin records (one name matches two rows) | 1 type | 2 types | 2 types |
| Monetary fields | $\sim$80% of domains | always | always |
| Fee formulas (payment recomputed by the tool) | — | — | 2–3 components |
| Multi-hop references (no shortcut fields) | — | — | 2–3 hops |

Table 10: **The difficulty ladder across the four self-generated releases.** The student (Qwen3.5-4B) attempted every task eight times. *In-band* is the share of tasks whose pass rate falls in $[0.1, 0.5]$, the range in which the eight rollouts of a task still disagree and a group-relative reward therefore carries a learning signal; *Ever* and *Always* are the shares of tasks solved in at least one and in all eight rollouts; *Mean pass* averages success over all rollouts. Releases glm_ds_0.3 and glm_ds_0.4 were produced by the same third version of the assignment generator in two separate waves with identical difficulty controls (Table 11); pooled, they form 983 tasks over 35 domains with 16.1% in-band, 73.0% ever solved, 33.3% always solved, and mean pass 0.57.

| Release | Version | Tasks (dom.) | In-band % | Ever % | Always % | Mean pass |
|---|---|---|---|---|---|---|
| glm_ds_0.1 | v1 | 30 (1) | 0.0 | 93.3 | 70.0 | 0.89 |
| glm_ds_0.2 | v2 | 84 (3) | 13.1 | 88.1 | 54.8 | 0.75 |
| glm_ds_0.3 | v3 | 561 (20) | 16.9 | 82.0 | 38.9 | 0.65 |
| glm_ds_0.4 | v3 | 422 (15) | 14.9 | 61.1 | 25.8 | 0.46 |

### A.9 TRAINING RECIPES

Table 12: Training recipes at 0.8B, 2B and 4B (p^1, %). Cold GRPO starts from the base model, SFT + GRPO from the SFT model; OPD teacher size in parentheses. Bold: best within a size; dashes: not available.

| Model | WorkBench | Workplace | BFCL | GAIA2 | $\tau^3$ |
|---|---|---|---|---|---|
| Qwen3.5-0.8B | 15.0 | 16.0 | 6.1 | **1.3** | **24.2** |
| + SFT | 23.4 | 24.2 | **13.0** | 0.6 | – |
| + cold GRPO | 10.0 | 22.2 | 5.9 | – | – |
| + SFT + GRPO | **27.0** | **25.9** | 8.8 | 1.0 | 21.6 |
| + OPD (2B teacher) | 21.2 | 23.9 | **13.0** | – | 13.2 |
| Qwen3.5-2B | 22.9 | 29.4 | 17.5 | 4.1 | 31.9 |
| + SFT | 38.8 | 42.8 | 24.6 | 3.2 | – |
| + cold GRPO | 39.5 | 47.6 | 26.4 | – | – |
| + SFT + GRPO | 43.0 | 46.6 | 25.3 | 4.5 | **37.8** |
| + OPD (4B teacher) | **49.5** | **50.4** | **31.3** | **7.4** | 30.5 |
| Qwen3.5-4B | 62.4 | 59.2 | 41.0 | 18.0 | 59.5 |
| + SFT | 63.4 | 64.0 | 44.3 | – | – |
| + cold GRPO | 70.3 | 73.8 | 47.4 | **27.0** | **65.3** |
| + OPD (4B teacher) | **70.7** | 73.1 | **50.9** | 23.5 | 62.7 |
| + OPD (9B teacher) | 67.7 | **75.6** | **50.9** | 21.5 | – |

Table 12 compares cold GRPO, SFT+GRPO and OPD at 0.8B, 2B and 4B, with base and SFT-only references. At 0.8B, cold GRPO lowers WorkBench from 15.0 to 10.0, while SFT followed by GRPO raises it to 27.0. At 2B both GRPO variants beat SFT alone. The capable 4B base already calls tools, so SFT on the trace corpus adds little on WorkBench (62.4 to 63.4), while GRPO from the base, our main 4B model, reaches 70.3. The same environments also support token-level supervision from previously trained expert models. We distill a 2B teacher into the 0.8B student and a 4B teacher into the 2B student, and compare 4B and 9B teachers for the 4B student. At 0.8B, OPD trails SFT+GRPO on WorkBench and Workplace (21.2 and 23.9 against 27.0 and 25.9) and leads it only on BFCL (13.0 against 8.8). The 2B OPD model improves all three execution benchmarks over SFT+GRPO (49.5, 50.4 and 31.3 against 43.0, 46.6 and 25.3). At 4B, both OPD students score higher than GRPO from the base on BFCL (50.9 against 47.4), and the 9B teacher gives the highest Workplace score (75.6). GRPO from the base keeps the highest GAIA2 score (27.0 against 23.5 and 21.5), so our main 4B model uses it. On $\tau^3$, OPD falls below the base at 0.8B (13.2 against 24.2) and at 2B (30.5 against 31.9), while SFT + GRPO reaches 37.8 at 2B; at 4B, GRPO from the base leads (65.3 against 62.7 for OPD). Corpus, teacher and training budget differ across recipes, so the table shows which recipes work at each size rather than isolating the objective.

### A.10 TRAINING CURVES FOR THE SELECTION ARMS

Reward and usable-group share over training, for the band-calibrated corpus against a random draw of the same size.

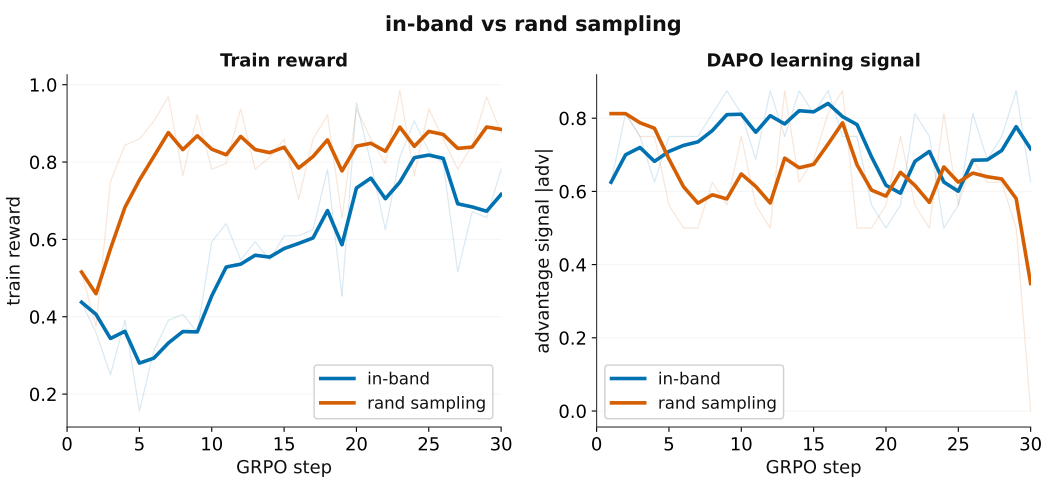



Figure 6: Training on the band-calibrated corpus against training on the same number of tasks drawn at random, over the same 30 steps. Left: mean episode reward per step. Right: the fraction of eight-rollout groups that are neither all-pass nor all-fail, which is the share of prompts that can move the policy at all. The random pool starts higher, because tasks the student already solves score well and teach nothing, and it carries fewer usable groups throughout. Both arms use the same base model, recipe and step budget.

### A.11 Reward hacking under cold GRPO on weak bases

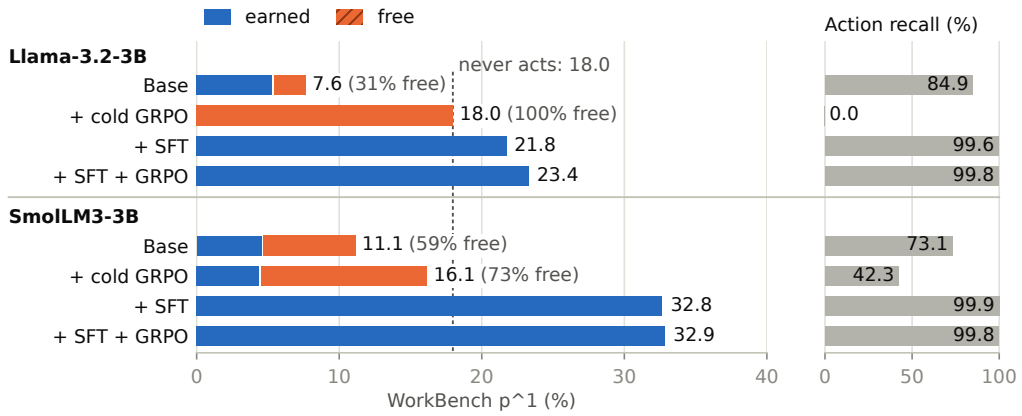



Figure 7: Left: WorkBench pass@k1 of Llama-3.2-3B and SmolLM3-3B, split into earned credit (the rollout calls a tool) and free credit (no tool call; possible only on the 18% of tasks whose gold leaves the database unchanged); dashed line: 18.0, the score of never calling a tool. Right: share of rollouts that call a tool on tasks that need an action.

A rising score can be a policy that stopped acting. A policy that never calls a tool scores 18.0 pass@k1 on WorkBench, because 122 of the 690 tasks (17.7%) have an empty gold and the seed state already satisfies two more. Figure 7 splits each score into the part a model earns with tool calls and the free part it scores without one, and reports action recall: the share of rollouts on tasks that need an action in which the policy calls a tool. Llama-3.2-3B under cold GRPO, trained with a partial-credit variant of the reward, climbs from 7.6 to 18.0, the never-act score: its action recall falls from 84.9% to zero and its whole score is free. SmolLM3-3B makes the same trade in part, with 73% of its

cold-GRPO score free and an action recall of $42.3\%$. SFT first closes this exit: the free part falls to $0.1$ points or less, and GRPO after SFT keeps it there. Llama-3.2-3B then rises from $21.8$ to $23.4$ and SmolLM3-3B from $32.8$ to $32.9$, and both call a tool in $99.8\%$ of the rollouts that need one. For this reason, we read the free share alongside every score on a weak base.

### A.12 Paired bootstrap over tasks

Table 13: Paired task-level bootstrap of each + Ours model of Table 2, and of the 0.8B SFT + GRPO model of Table 12, against its base: trained minus base in percentage points, with 95% percentile intervals from 20,000 resamples of tasks (GAIA2 capabilities, $\tau^3$ domains and BFCL categories are resampled separately and weighted equally). $\Delta$ is computed from unrounded scores, so it can differ by 0.1 from the difference of the rounded table entries; the $\tau^3$ $\Delta$ uses the tasks valid for both models. The intervals cover the sampling of tasks within one evaluation, not variation across training runs. Tasks: scored for both models. $k = 3$ for WorkBench and GAIA2, $k = 5$ for Workplace and $\tau^3$. Dashes: not available.

| Model | Benchmark | Tasks | $\Delta$ p^1 [95% CI] | $\Delta$ p^k [95% CI] |
|---|---|---|---|---|
| Qwen3.5-0.8B | WorkBench | 690 | $+11.9$ [$+9.5$, $+14.4$] | $+6.2$ [$+3.6$, $+8.8$] |
| + SFT + GRPO | Workplace | 545 | $+9.9$ [$+7.9$, $+12.0$] | $+5.7$ [$+3.1$, $+8.4$] |
| | BFCL | 800 | $+2.6$ [$+0.6$, $+4.8$] | |
| | GAIA2 | 800 | $-0.4$ [$-1.0$, $+0.2$] | 0.0 [0.0, 0.0] |
| | $\tau^3$ | 273 | $-2.2$ [$-5.1$, $+0.6$] | $-0.6$ [$-3.2$, $+2.1$] |
| | ToolSandbox | 509 | $-7.2$ [$-10.6$, $-3.8$] | |
| Qwen3.5-2B | WorkBench | 690 | $+26.5$ [$+23.5$, $+29.6$] | $+23.2$ [$+19.6$, $+26.8$] |
| + Ours | Workplace | 545 | $+21.0$ [$+18.2$, $+23.8$] | $+20.9$ [$+17.2$, $+24.6$] |
| | BFCL | 800 | $+13.8$ [$+10.5$, $+16.9$] | |
| | GAIA2 | – | – | – |
| | $\tau^3$ | 275 | $-1.3$ [$-5.0$, $+2.5$] | $-2.6$ [$-5.3$, $+0.2$] |
| | ToolSandbox | 509 | $+4.2$ [$+1.0$, $+7.5$] | |
| Qwen3.5-4B | WorkBench | 690 | $+8.0$ [$+5.8$, $+10.1$] | $+9.0$ [$+5.5$, $+12.5$] |
| + Ours | Workplace | 545 | $+14.6$ [$+12.2$, $+17.0$] | $+16.0$ [$+12.1$, $+20.0$] |
| | BFCL | 800 | $+6.4$ [$+3.3$, $+9.5$] | |
| | GAIA2 | 800 | $+9.0$ [$+7.2$, $+10.8$] | $+6.9$ [$+4.9$, $+8.9$] |
| | $\tau^3$ | 277 | $+5.9$ [$+2.2$, $+9.7$] | $+7.7$ [$+1.9$, $+13.6$] |
| | ToolSandbox | 509 | $+6.0$ [$+3.1$, $+9.0$] | |
| Qwen3.5-9B | WorkBench | 690 | $+1.4$ [$-0.4$, $+3.2$] | $+3.2$ [$+0.4$, $+6.1$] |
| + Ours | Workplace | 545 | $+7.0$ [$+5.1$, $+9.0$] | $+10.1$ [$+6.4$, $+13.8$] |
| | BFCL | 800 | $+0.9$ [$-1.6$, $+3.4$] | |
| | GAIA2 | 800 | $+4.1$ [$+2.3$, $+6.0$] | $+4.0$ [$+1.5$, $+6.4$] |
| | $\tau^3$ | – | – | – |
| | ToolSandbox | 509 | $+1.8$ [$-0.8$, $+4.4$] | |

### A.13 Training configuration

Every optimiser, rollout and reward setting the runs read, taken from the launchers and the frozen per-run configs.

### A.14 Examples of in-band tasks

Figure 8 shows four tasks from the calibrated corpus. Each sits inside the learning band for the student, which solves it in some of its eight attempts but not all, while the teacher solves it more often, so the tasks are hard rather than broken (Section 3.2). The four examples cover different sources of difficulty: a chain of lookups ending in a computed fee, an ambiguous reference that requires a clarification, a filter whose every match needs its own write, and a conditional branch decided by a field of the record.

Table 14: Training configuration. The first column covers the GRPO runs of the 4B and 9B models of Table 2, which share one recipe; the SFT column is the single corpus that produced both small-tier initialisations; the last column is GRPO started from those. A dash means the row does not apply to that stage. Values come from the launchers, frozen configs and saved checkpoints of the runs. The launcher of the first column leaves the optimiser, clipping and KL settings at the framework defaults, so those rows are read from the frozen config of an earlier run through the same launcher. The reward weights are listed because the launcher exports them, not because they act.

| | GRPO from the base (4B and 9B) | SFT (2B and 0.8B) | GRPO from SFT |
|---|---|---|---|
| *model* | | | |
| base / adaptation | Qwen3.5-4B and 9B; LoRA $r$=32, $\alpha$=64, all-linear | Qwen3.5-2B and 0.8B; LoRA $r$=32/64 | the SFT checkpoint; LoRA $r$=32/64 |
| *optimiser* | | | |
| lr / schedule | 1.5e−5, constant, no warmup | 1e−4, constant | 5e−6, constant |
| AdamW | (0.9, 0.999), wd 0.01, grad-clip 1.0 | same | same |
| *batching* | | | |
| batch / length | 8 prompts × 8 rollouts = 64 episodes/step; 2 epochs | bs 1, grad-accum 4; 1 epoch, 245 steps | same |
| *rollout* | | | |
| sampling / context | $T$=1.0, top-$p$ 1; 6144 prompt / 14336 response; 20 assistant turns | 32768 total | $T$=0.7; 6144 / 14336; 25 turns |
| engine | vLLM async, TP 1, prefix-caching off; 9B: FSDP parameter and optimiser offload | — | same |
| *objective* | | | |
| estimator / aggregation | GRPO, advantages not std-normalised, `seq-mean-token-mean` | cross-entropy, assistant-masked | same as GRPO from the base |
| clip / KL | $\epsilon$=0.2, dual-clip 3.0, 1 PPO epoch; no KL loss, no KL in reward | — | same |
| *reward* | | | |
| verdict | strict conjunction over ten terms, 1 or 0; no judge, no partial credit; the calibration reads the same verdict | — | same |
| penalties | forbidden call 1.0; repeat call 0.15, capped 0.30 | — | same |
| *cost* | | | |
| hardware | 4B: 4×H100; 9B: 8×H100, ≈4.5 h for 60 steps | 1 GPU, 12 min | 4×H100 |

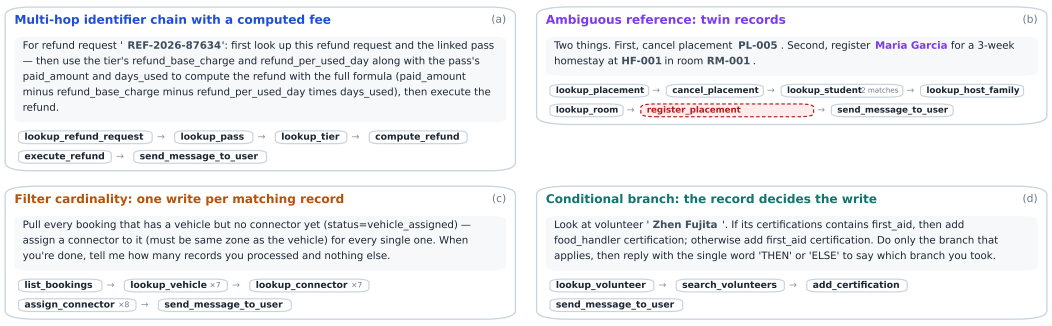



Figure 8: Four in-band tasks from the generated corpus. Student and teacher pass rates over eight attempts are (a) 0.38/0.75, (b) 0.25/0.50, (c) 0.38/1.00 and (d) 0.12/1.00. All four requests are template instances; the highlighted values are substituted at generation time. (a) A multi-hop identifier chain with a computed fee: the refund is reachable only through chained lookups, and the paying tool recomputes the fee and rejects a mismatched amount. (b) An ambiguous reference: two students share the queried name, so the agent completes the unambiguous half, and a write to either twin is forbidden until the user answers. (c) Filter cardinality: every booking that matches the filter needs its own write, and the report is due only after the last one. (d) A conditional branch: the certifications recorded on the volunteer's file decide which of the two writes is legal.