SKILLTRAINBENCH: CAN LLM AGENTS AUTHOR SKILLS THAT IMPROVE OTHER LLM AGENTS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Skills are compact packages of instructions and reference material supplied to an LLM agent's harness, allowing its behaviour to change without updating model weights. We ask whether LLM agents can write such skills for other LLM agents. SkillTrainBench gives a curator agent a fixed interaction budget with one or more learner agents whose model weights remain fixed. The curator iteratively writes and revises a skill, chooses which learner-task pairs to probe, and uses the resulting performance to guide further revisions. The final skill is evaluated on private held-out tasks. We study four domains drawn from HealthBench, Humanity's Last Exam, quantitative-finance evaluations, and τ^3 Banking, using multiple frontier curators and learners spanning a range of capabilities. We find that curator-written skills improve every learner on HealthBench and transfer to models that never saw them, but rarely help on the other three domains. Skill quality is also non-monotonic: performance frequently peaks before the final iteration, and curators rarely submit their best skill because the development scores they see cannot rank their own drafts. SkillTrainBench therefore provides a controlled setting for studying how LLM agents can improve other LLM agents through externally authored skills, and where this process currently fails.

1 INTRODUCTION

Written instructions can change how a language model agent works without changing its weights. These instructions include prompts, tool descriptions and *skills*, packages of instructions and reference material that an agent reads during a task. We call the agent writing a skill the curator and its recipient the learner. We ask whether an agent using one model can write skills that improve an agent using another, and how reliably it identifies its best version (fig. 1). This question matters to teams with only API access to a model, labs shipping models with skills and platforms configuring subagents: all can change an agent's context after its weights are fixed.

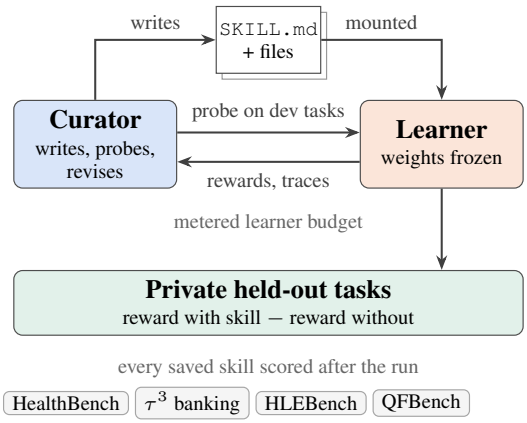

Figure 1: *A curator writes a skill for a frozen learner, tests it on development tasks under a metered budget, and saves versions scored against no skill on private held-out tasks in four domains.*

Existing evidence does not establish whether agents can write effective skills for other agents. Human-curated packages improve performance across models and harnesses (Li et al., 2026). PostTrainBench tests agent-led improvement through weight updates (Rank et al., 2026), whereas agents' self-written skills can reduce performance (Li et al., 2026) and generated skills lag behind human-authored ones (Zhong et al., 2026).

Measuring skill authorship requires separating learning from artefacts of evaluation. Private tasks prevent a curator from copying answers or grading criteria into a skill. A content-free placebo

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.

controls for the effect of delivering a file. Checking whether the learner opens the skill helps interpret null results, since weaker models often fail to load the skills they are given (Lin et al., 2026).

SKILLTRAINBENCH measures changes in a frozen learner's reward from curator-written skills (sections 3 and 4.1). The curator selects development tasks, tests the learner through a gateway that meters its tokens, reads attempt traces and revises its skill. We score every saved version against the same learner without a skill on private held-out tasks, then give submitted skills to learners that never saw them.

The clearest gains occur on HealthBench clinical consultation: across three frontier curators, skills improve all three learners and four further models that do not see those skills during curation (section 5). Across the four domains, the same curators rarely improve held-out reward on banking dialogue, expert science or quantitative finance. In every domain, curators save better versions than they submit: development-task scores do not reliably rank their drafts (section 6).

2 RELATED WORK

Improving a frozen model without training. PostTrainBench measures how well agents improve a model by updating its weights under a fixed compute budget (Rank et al., 2026). We keep the weights fixed and measure what an agent can achieve by adding text to its context. Related work shows that a stronger model can guide a weaker one through explanations (Saha et al., 2023), skills (Jiang & Huang, 2026; Shi et al., 2026) or a harness refined on a validation split (Qian et al., 2026). Each study proposes a method and reports its gain on public benchmarks; AI4AI, like our curator, iterates against a split it can see. Our benchmark instead evaluates the author itself: it probes a separate learner through a metered service, and every skill it writes is scored on private held-out tasks.

Evaluating skills. SkillsBench scores curated skill packages across model and harness configurations. It finds that skills the agents write for themselves can lower performance (Li et al., 2026). Generated and self-evolved skills have their own benchmarks (Zhou et al., 2026; Lei et al., 2026; Gao et al., 2026) and fall short of human-authored ones (Zhong et al., 2026). A skill can help some task types and hurt others (Wang et al., 2026). Paired trials measure its lift (Kevin et al., 2026), and whether it is loaded at all differs between models (Lin et al., 2026). Our benchmark tests the skill author: an agent chooses probes on public tasks, while a separate frozen learner is scored on private tasks. Every saved skill is evaluated against the skillless baseline and a placebo.

Methods that write and revise skills and harnesses. Existing methods revise skills using verifier feedback (Liu et al., 2026), model diagnostics of agent runs (Gautam et al., 2026) or held-out validation scores (Yang et al., 2026), and evolve harnesses around a fixed model (Wei et al., 2026; Jiang et al., 2026), and EvoHarnessBench tests agents as their harness changes (Ke et al., 2026). The revision and evolution methods are candidate curators rather than competing benchmarks, and each can run through our curator interface.

Prompt optimisation and experience memory. Prompt optimisers tune prompts or demonstrations for a fixed task model against a task metric (Zhou et al., 2023; Khattab et al., 2024; Yang et al., 2024; Opsahl-Ong et al., 2024; Yuksekgonul et al., 2024; Agrawal et al., 2026), and memory methods reuse earlier attempts, mostly for the model that made them (Shinn et al., 2023; Zhang et al., 2026; Suzgun et al., 2026; Wang et al., 2025; Zhao et al., 2024; Wang et al., 2024). Our curator chooses probes on public tasks and writes a skill package with multiple files for another model. The recipient is scored only on private tasks. Most of these methods require an adapter for each combination of harness, model and benchmark before they can use our curator interface.

3 PROBLEM SETTING

3.1 SKILL AUTHORING

The curator C writes and revises a skill s, a set of text files for the frozen learner L, by testing on public development tasks $\mathcal{D}$. Each agent pairs a model with software that runs it, called a harness. C

reads bounded learner traces but cannot see private held-out tasks $\mathcal{H}$. L cannot edit s. Curation stops at submission or when a limit in $\mathbf{b} = (B, K, c, T)$ binds. The gateway enforces learner token budget B across tests and meters curator calls separately without refusing them. K caps tests on bounded batches, c caps turns per learner attempt, with cap hits scored as failures, and T caps curator time. The binding limit varies by run.

3.2 SCORING

We measure mean reward on the same held-out tasks with no skill, skill s, or a content-free placebo p matched in layout and delivery:

$$r(a) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} R(h; L, a, \xi_h), \qquad a \in \{\emptyset, p, s\}.$$

Verifier reward R scores learner L on task h with skill condition a and randomness ξ_h from sampling, simulated users or judges. HLEBench judges against reference answers, while τ^3 banking checks dialogue outcomes. Both give pass or fail scores. HealthBench scores the share of available positive rubric points met. QFBench gives pass or fail scores except on 14 of 86 tasks that allow partial credit. Timeouts and turn-cap hits score 0. Model API samples cannot be repeated, so each condition draws fresh ξ_h.

Every run reports gross lift, $\Delta_{\text{gross}} = r(s) - r(\emptyset)$, the gain over no skill, including any benefit of receiving a file. Net lift, $\Delta_{\text{net}} = r(s) - r(p)$, estimates instruction effects against the placebo, though content and packaging may interact. We measure net lift with one placebo evaluation per learner and domain, shared by all curators.

We report uncertainty for runs with at least twelve distinct held-out tasks using a 95% percentile interval from $20{,}000$ paired bootstrap resamples of tasks, each represented by its mean trial reward. The interval covers task and learner variation for one skill, but not variation between curator runs, which our replications suggest is of similar size. The skill is the only controlled difference across independent evaluations with identical learner, tools, initial state, limits and verifier.

4 EXPERIMENTAL SETUP

Roles. The curator and learner deliberately use asymmetric configurations. For example, the curator uses a current frontier model under its native harness: `claude-opus-5` running `claude code` (the `claude-cli`, with its version pinned in the sandbox image), acting only through six MCP tools: `list_dev`, `show_dev`, `read_trace`, `write`, `probe`, and `submit`. The goal is to give the curator the strongest configuration for writing skills so that we can measure how well a skill improves agent performance on a hidden held-out dataset. The learner, in contrast, uses a less powerful model, such as `claude-sonnet-4-6` under `claude code`. The learner configuration remains unchanged during probing and held-out evaluation. This asymmetry makes the measurement viable: a powerful learner such as `claude-opus-5` or `claude-fable-5` drives benchmark baselines towards saturation, leaving little room for skills to demonstrate an improvement. A weaker learner, however, leaves baselines contested on hard tasks. When the evaluator mounts the curated skill on held-out tasks, a genuine teaching effect is therefore expected to appear as a change in the performance score rather than as noise near a ceiling.

Curation and evaluation workflow. The workflow proceeds as follows (fig. 2). All skill generation happens inside a single orchestrator sandbox. The curator edits the skill files; a probe then runs the frozen learner on a development subset that the curator randomly selects from a training pool. A metering gateway sits between the curator and the probed learner: every learner call passes through it, and it enforces the learner token pool. Each probe result reports the tokens remaining in the pool; this readout is the curator's only view of its budget, which it uses to decide how many tasks to include in the next probe and how many further iterations to run. The curator generates a skill by reading trajectories of tasks that fail in a probe. The live skill is then snapshotted and applied to all tasks in the next probe. When curation ends (on pool exhaustion, an iteration cap, or curator submission), the run holds $N+2$ snapshots: one per probe iteration, the final skill, and a `baseline` folder containing no skill. The orchestrator then syncs the snapshots to the artefact store, dispatches

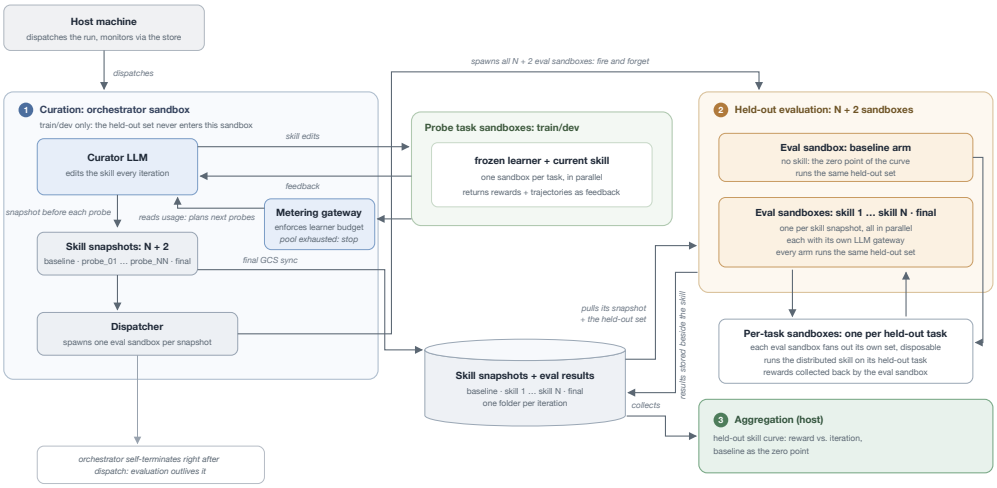

Figure 2: The deferred evaluation pipeline. A single orchestrator sandbox runs curation and snapshots the skill at each iteration. It then dispatches one evaluation sandbox per snapshot and terminates.

one evaluation sandbox per snapshot ($N+2$ concurrent sandboxes), and terminates. Each evaluation sandbox is equipped with the skill from one iteration, pulls the same held-out task set, and starts its own LLM gateway. The evaluation workload runs outside the evaluation sandbox: each sandbox distributes the held-out tasks and skill to independent, disposable per-task sandboxes whose only network access is that gateway, then collects the per-trial verifier rewards.

Isolation. The isolation measures address cheating by the curator: a frontier model pressed to produce the best possible skill could look up benchmark answers (on the web or in the local dataset) and put them into the skill, which could then score without teaching. Three mechanisms mitigate this problem. *(i) A closed tool surface.* The curator interacts with its environment only through the six MCP tools listed above (`list_dev`, `show_dev`, `read_trace`, `write`, `probe`, and `submit`), and its native online search tool is disabled in the harness configuration. *(ii) Logged model traffic.* Every call between the curator and its model passes through the metering gateway, which appends a trace of each call to `curator_transcript.json`. The orchestrator treats the curator as the sequence of calls recorded there, so evasive actions (probing the tool surface for gaps or reasoning towards the held-out set) leave traces, and we scan the transcript at the end of each run. We monitor potential cheating through these logs rather than by trusting the model. *(iii) A separate evaluation sandbox.* The held-out dataset never resides on the curator's machine: curation runs in the orchestrator sandbox, while a separate evaluation sandbox hosts the held-out set and exposes an endpoint that the curator's sandbox calls to trigger evaluation. As a result, the curator has no access to the held-out data.

Learner execution and turn limits. The learner executes inside the Harbor agent framework, with one disposable sandbox per task, digest-pinned task images, and network egress restricted to the metering gateway. Within a run, the learner is frozen: its (model, harness) pair is fixed at dispatch and hashed into a run manifest written before the first model call; the curator's only interface to it is the probe service on development tasks, and the curator cannot edit the learner; the gateway ledger stamps every curator call. Probes differ only in the mounted skill directory. We also cap the number of turns a learner may take on a single task. The dry baseline runs motivate this cap: attempts that fail to converge within a benchmark's natural horizon often end in an agent timeout. Without turn limits, a task that does not terminate consumes a disproportionate share of the learner budget pool. Therefore, an attempt that reaches the cap stops and scores as a failure. This conserves the pool for the remaining tasks.

Table 1: *Pinned learner limits. Split is development/held-out tasks. The turn cutoff per attempt is* `max_turns` *for Claude Code (CC) learners and* `max_iterations` *for OpenHands (OH) learners.*

Benchmark	Split	Learner pool	Probe calls × max tasks	Turns (CC)	Turns (OH)
HealthBench	200/160	80M	30×40	11	12
τ^3-bench banking	67/30	360M	30×40	51	60
HLEBench	412/105	400M	15×60	36	40
QFBench	54/14	160M	15×13	41	45
	total	1,000M			

Token budgets. The gateway enforces separate curator, learner probe, held-out evaluation and, where needed, LLM judge token pools, with 500M allowances for both the curator and evaluation. The learner pools in table 1 are rules of thumb based on mean tokens per task in dry baseline runs, the planned probe volume, and a fixed 1,000M total across the four benchmarks, as detailed in section C. Each probe reports the remaining learner balance so the curator can choose its next batch and number of probes; exhaustion ends curation with submission of the final skill.

4.1 Choosing the Domains

A domain qualifies when the learner scores well above zero without a skill but below the ceiling. Rulegame, coding and box-counting do not qualify because each development task hides a rule that all held-out tasks require, leaving a zero baseline. Gains there measure rule transmission, so these benchmarks only check the pipeline. Four qualifying domains use distinct verifiers to limit exploits, with task counts given in table 1. QFBench checks finance answers (CC BY-NC 4.0), HealthBench uses a model-graded rubric (MIT), and τ^3-bench banking checks simulated-user dialogue against database state (MIT). HLEBench judges short answers from Humanity's Last Exam in chemistry, engineering, and computer science and AI, where tools help (MIT, gated access). HealthBench grades each rubric item with OpenAI's official `simple-evals` prompt pinned to one commit and `claude-haiku-4-5-20251001` at temperature 0, replacing the official `gpt-4.1` at 0.5. HLEBench replaces the official `o3-mini` with `claude-sonnet-4-6`, so the same model grades Sonnet learner answers. Skillless baselines differ from published scores (section G) because of fixed harnesses and turn limits, no web access and subsets selected for room to improve. Timeouts count as failures. Each gain has a paired baseline under these conditions. The curator selects batches of development tasks for skill tests. The probe service knows only development identifiers and hides held-out identifiers from the curator. It rejects unknown or duplicate identifiers and oversized batches.

5 Results

A *cell* pairs one curator, learner and domain. In each of the 36 cells, we ran one curation at the canonical split and learner token budget, then froze and evaluated every saved skill, the submitted skill and the learner without a skill on the same private held-out tasks. The cell's *gain* is the submitted skill's mean reward minus the mean without it, paired by task. A paired task bootstrap gives its 95% interval. Table 20 reports all baselines, intervals and caveats.

5.1 Skills Help on One Domain and Rarely on the Others

On HealthBench all nine curator and learner combinations improved by $+0.09$ to $+0.28$, with intervals excluding zero. Averaged over curators, Sonnet 4.6 gained 0.18 from a baseline near 0.14, GLM 5.3 Flash gained 0.15 from about 0.27, and Opus 5 gained 0.11 from about 0.40: the lower the learner's baseline, the larger its gain. Opus 5 wrote the strongest skill for both Claude learners. The curators were indistinguishable for GLM. The weakest learner's HealthBench skill made it answer rather than defer to a clinician (section 6.3).

The other three domains show no consistent gain. Across 105 HLEBench questions, eight gains range from -0.03 to $+0.07$ and no interval excludes zero, even after combining curator runs. Across

30 τ^3 banking dialogues, intervals span about 0.3 and only one of nine excludes zero: GPT-6-sol's skill lifts GLM from 0.00 to 0.13. GLM often ends with a plain-text turn, which the environment scores as failure. Across 14 QFBench tasks, even effects of 0.2 have wide intervals. Opus 5's and Fable 5's skills lower GLM by about 0.2. Only GPT-6-sol clearly helps, by +0.17 for Sonnet 4.6 and +0.03 for Opus 5. These results show that current curators do not teach reliably on these domains, although human-written skills raised pass rates on related SkillsBench tasks in our pilot (section 5.3).

The placebo separates what a skill says from the effect of receiving any skill at all (section 3). It is a generic "expert task playbook", a checklist without domain content whose description invites the learner to use it on every task. Opus 5 opens it in nearly every attempt, and Sonnet 4.6 in most HLEBench and QFBench attempts but rarely on HealthBench or τ^3 banking. Where it is read, the checklist moves the score by at most about five points in either direction. The curated HealthBench skills beat it: for Opus 5, which reads the placebo on almost every task, two of three net lifts exclude zero. On HLEBench the reverse holds for Sonnet 4.6, whose three submitted skills all score below the generic checklist. For GLM we have no complete placebo evaluation on HealthBench or τ^3 banking.

5.2 The Curator Cannot Tell Which of Its Skills Is Best

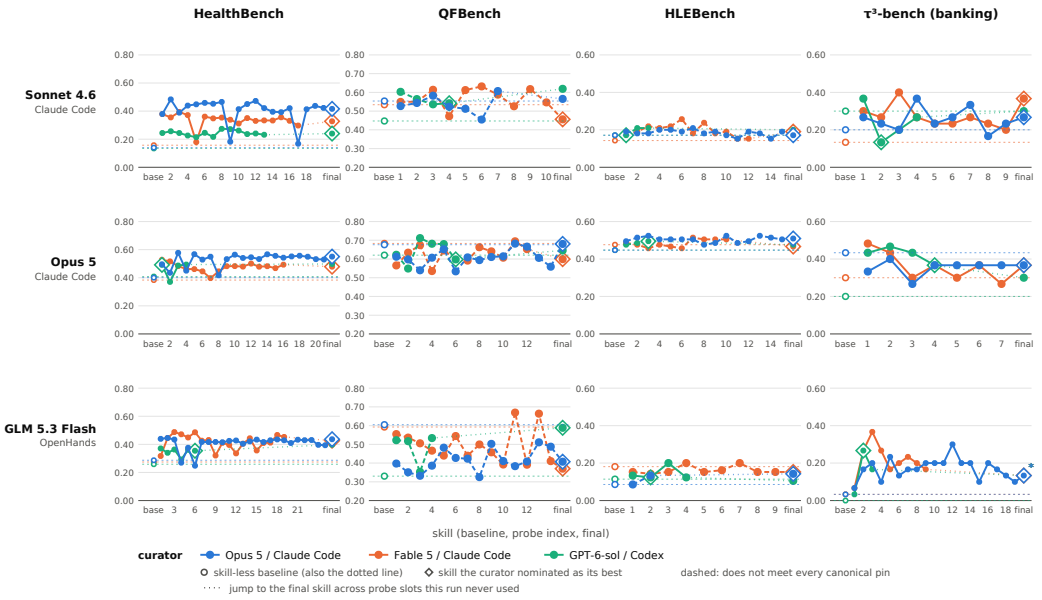

Figure 3: *Held-out reward for every saved skill, ordered by creation, with domains in columns, learners in rows and colours denoting curators. Open circles and dotted lines mark the no-skill baseline, diamonds the submitted skill, and dashed lines evaluations without a recorded turn cutoff.*

Held-out reward across saved skills fluctuates during curation (fig. 3). It falls between consecutive skills in 35 of 36 runs, five times per run on average, and the best skill precedes the final one in 31 runs. On HealthBench the first skill already provides most of the gain. Later versions raise or lower it. The curator's declared best skill achieved the highest held-out reward in only 4 of 35 declaring runs.

Curators chose different versions but rarely chose the held-out best. Opus 5 and Fable 5 declared their final skill in all 23 runs. GPT-6-sol chose an earlier version favoured by its development evaluations in 10 of 12 runs. In its τ^3 run with GLM, it discounted a later evaluation because of "six infrastructure failures" and returned to the last skill it trusted. Yet its choice was best on held-out tasks in only 3 of 12 runs. Section 6.2 explains why development evaluation scores cannot support reliable selection.

5.3 The Human Reference

SkillsBench offers a human-written reference: its 87 tasks have skills written independently of the benchmark or selected from public repositories, each covering a problem class rather than one answer (Li et al., 2026). Across 18 agent configurations, these skills raise pass rates from 33.9% to 50.5%. We compare no skill, the task's SkillsBench skills and our curator's final skill for the same learner, matching finance and economics to QFBench, natural science to HLEBench and, weakly, one clinical laboratory task to HealthBench (section F). This comparison favours human skills: they match each task's problem class and their descriptions identify it, helping Claude Code decide to open them, whereas our skills were written for another benchmark in the same field.

In a ten-task pilot with one trial per learner and task (table 21), Sonnet 4.6 passed 3, 7 and 1 tasks with no skill, human skills and our package. Opus 5 passed 2, 8 and 2. Learners opened human skills in 18 of 20 trials and ours in 3 of 20, so our package usually served as a second baseline. The pilot therefore suggests that the human skills help on these tasks, but it cannot compare their content with ours.

5.4 Transfer to Learners That Never Saw the Skill

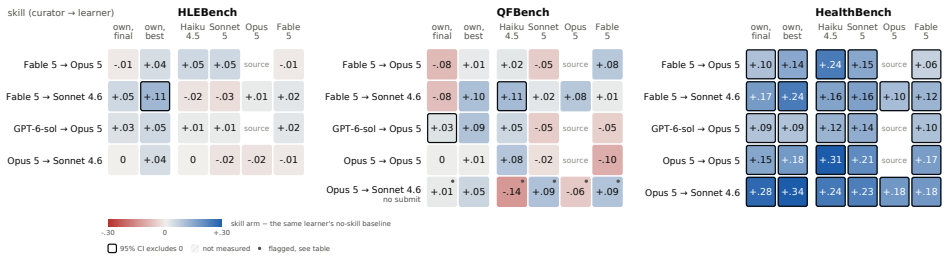

Figure 4: *Each frozen submitted skill was given to learners that had never seen it. Colour shows gain over each learner's no-skill baseline. Reference columns show the original learner with the submitted skill (*own, final*) and its highest-scoring saved skill on held-out tasks (*own, best*). The curator never saw those scores, so the latter is an optimistic reference. Only runs at the canonical settings appear. HLEBench and QFBench lack a recorded turn cutoff. Heavy borders mark* 95% *intervals excluding zero, hatching unmeasured pairs, and dots caveats in table 20.*

We tested whether submitted skills help learners for which they were not written by giving each skill unchanged to other Claude models, from Haiku 4.5 to Fable 5. Each target was compared with its own no-skill baseline on the same held-out tasks (fig. 4).

On HealthBench five skills from all three curators improved every target learner: all 17 intervals lie above zero. Opus 5's skill for Sonnet 4.6 raised Haiku 4.5 by 0.24, Sonnet 5 by 0.23, and Opus 5 and Fable 5 by 0.18 each. Gains almost always decrease as baseline reward rises, from Haiku 4.5 near zero to the highest-scoring learners, Opus 5 and Fable 5. On HLEBench and QFBench, skills rarely helped their original learners and almost every transfer interval crosses zero, leaving transfer inconclusive. The HealthBench results show skills that help across models, not just the learner for which they were written (section H).

6 What the Curators Actually Did

6.1 How a Curation Run Unfolds

A curator receives a brief, sample development tasks and six tools to list and read tasks, write its skill, probe the learner and submit. A `probe` runs the frozen learner once on each development task the curator names, with the current skill, and returns every reward, a trace identifier for each attempt and the remaining token budget. With `read_trace` the curator can read up to 40 such traces per run. Each probe also saves the skill as it stood, and these saved skills are what fig. 3 evaluates. A run ends at submission or when tokens, probe calls or time run out.

Table 2: *Curator traces record behaviour across* 12 *cells each. Values are medians except* reverted*, the count selecting an earlier skill, and* gain*, mean held-out gain.* Pool *gives the share of learner tokens spent.*

Curator	Probes	Pool	Trace reads	Writes	Versions	Reverted	Gain
Opus 5 (Claude Code)	14.5	85%	27	20.5	18	0 of 11	+0.053
Fable 5 (Claude Code)	11	65%	28.5	12.5	11	0 of 12	+0.032
GPT-6-sol (Codex)	4	10%	32	4	4	10 of 12	+0.089

All three curators work in the same loop of probing, reading the traces of failed attempts and revising, but at very different intensities (table 2). Opus 5 spends about three and a half hours making 14 to 15 probes of 24 to 30 tasks, uses 85% of learner tokens and rewrites about 180 lines per probe. Fable 5 takes as long but makes 11 probes and edits about 50 lines per probe. GPT-6-sol takes about an hour, makes four probes of about 20 tasks, uses a tenth of the tokens and changes about 25 lines per probe, yet reads as many traces. More probing did not yield more gain: GPT-6-sol has the highest mean gain and improves all three QFBench learners, whereas Fable 5 lowers all three. Each curator runs in its own harness, so this comparison cannot rank the models. All three undo changes that a probe shows to be harmful. Opus 5 removed its "probe-20 additions" after neither target task improved, and GPT-6-sol restored a better tested skill after infrastructure failures. Section J compares their strategies.

6.2 Why the Curator Cannot Pick Its Best Skill

The selection failure of section 5.2 has a measurable cause: the development scores a curator sees do not rank its skills within a run. Across all 370 probes, development and held-out scores do correlate (Spearman 0.67), but only because some benchmarks and learners are harder than others. Within runs, mean Spearman correlation is -0.06 (95% interval $[-0.22, +0.10]$); probes on identical development batches agree with the held-out ranking in only 56% of pairs. Choosing the highest development score would give 0.376 mean held-out reward versus 0.375 for submitted skills and select the best in 2 of 36 runs. Choosing each run's best skill with hindsight would add 0.069 $[0.048, 0.093]$. The same development task and skill yield rewards with standard deviation 0.21, so batches of 20 to 40 tasks cannot distinguish gains of a few points. Selection needs larger or repeated probes.

6.3 What the Skills Contain

Submitted skills combine domain advice with tool instructions. One HealthBench skill warns about "under-coverage" and lists emergencies requiring referral. A QFBench skill urges an early runnable script and notes different defaults in `pd.Series.std` and `np.std`. A τ^3 skill says "Every turn must contain a tool call. Your prose is invisible." Domain knowledge accounts for 39% to 64% of words, while tool use reaches 48% in τ^3 banking. None names its learner. Across all four domains, one curator's skills for different learners are more similar than different curators' skills for one learner (mean cosine similarity 0.36 versus 0.29). This domain focus is consistent with HealthBench transfer (section 5.4). The skills do not copy the evaluation grader: their few five-word matches with HealthBench rubric criteria are generic. No skill matches an HLEBench answer or QFBench verifier literal. Longer skills yield no greater gains.

The largest HealthBench gain comes from Sonnet 4.6 answering questions it previously deferred to a clinician. Without a skill, its complete, well formed answers are brief and cautious, giving it the lowest baseline. With the skill, answers grow longer and score about twice as high at equal length. Length explains less than a fifth of the gain; context awareness and accuracy improve most.

6.4 Whether the Learner Reads the Skill

We study two learner harnesses, Claude Code and OpenHands, and they give learners different access to a skill. Claude Code shows a one-line description and lets the learner choose whether to open `SKILL.md`, which it did in 1,877 of 2,589 held-out attempts, including 145 of 180 on τ^3 banking. Sonnet 4.6 never opened GPT-6-sol's τ^3 skill in 30 attempts and gained nothing. OpenHands inserts

`SKILL.md` into the prompt, but learners opened another file of the skill in only 14 of 927 attempts, so a second file is mostly unread. Opening the skill is not enough for a gain: across the Claude Code evaluations the opening rate barely correlates with the gain (Pearson 0.23), and on the same task, attempts that opened the skill did no better.

6.5 Failures of the Scaffold, Not of the Agent

Some low scores and missing submissions come from the infrastructure, not from the curator or the learner, and we handle them by fixed rules. Some curators stopped before submitting because a model stream stalled, the model refused or a watchdog stopped the harness. In these cases we score their last skill only if the skill and all its evaluations are complete, and exclude placeholder skills; the last skill was not necessarily the curator's choice (section 5.2). We exclude runs dominated by an endpoint outage because the resulting timeouts score zero. Learners can also use nearly their entire budget retrying a malformed tool call (section I). Timeouts that nevertheless produced an answer remain in the mean. On QFBench, with 14 tasks, two timeouts move a mean by more than the effects we report. Every run records why it was included or excluded.

7 Discussion and Conclusion

An agent can improve another by writing a skill, but the effect depends on the domain. On HealthBench, every curator improves every learner, and all five transferred skills improve four unseen models, especially those with the lowest baselines. On the other three domains, the same curators rarely change held-out reward. Across domains, curators produce better intermediate skills than those they submit because development scores do not reliably rank their drafts.

The transfer result matters to teams that access models through an API and labs that ship models with skills: a skill written by a strong model improves weaker and cheaper models without training. The selection result limits current automation. Curators fail to submit their best skills when development scores are noisy, a weakness shared by any loop that selects a prompt, skill or harness using a small batch.

Limitations. Our intervals cover tasks and learner samples but not variation between curations. We have one curation per cell; the HLEBench repeats vary about as much as the configurations. Our grader and baselines differ from the official protocols by design (section G). Some HLEBench and QFBench evaluations do not record or apply the turn cutoff, and their curators have 15 probe calls rather than 30 (tables 19 and 20). The placebo has no baseline from the same run, and we score one skill per learner, leaving skill choice and interactions unresolved.

Open questions. The central open question is whether better evidence helps curators submit their best skill. Larger or repeated probes, or reserved development tasks, could rank drafts more reliably. Repeated curations could estimate variation within each cell, while the full human reference (section 5.3) could express each gain as a share of expert performance. We release the benchmark to study these questions, skill libraries, executable skills and whether one agent's context can steer another away from its task.

AI use statement

(This section is **required** and does not count toward the page limit.)

We use generative AI to implement benchmark methods and analysis, assess experimental methodology, analyse curator traces, interpret results, and clean and reformat task datasets. We use Claude Opus 5, Opus 5.5 and Claude Fable 5 through the Claude Code coding assistant, and OpenAI GPT-6-sol and GPT-6-astra through the Codex CLI. We do not use generative AI to generate synthetic data (all tasks come from published benchmarks), develop theoretical models, formulate or prove mathematical claims, or translate text.

We also use these tools to create and edit code and artefacts, create figures and tables, draft and edit the paper for readability, search and summarise literature, audit citations against cited papers, and format references.

The repository's scoring code computes every reported number from stored run artefacts, and we cross-check them against independently computed values. The authors review and test the code, check every citation against the cited paper, and review all text.

Generative AI is also the object of study here, not only a tool: the curator and learner agents are the system under measurement, and their prompts and budgets are specified in section 4.1 and section D.

We take responsibility for the final content of this work, including text, claims and artefacts produced with generative AI.

Ethics statement

The benchmark uses published task sets and involves no human subjects or private data. HealthBench contains medical consultations, but we score model replies against the published rubric, make no clinical claim, and do not intend any output for care. Agents run in isolated containers with egress restricted to a metered gateway, so a curator cannot reach other runs or the public internet in the closed track. Token caps limit curator and learner spending, and a broker key imposes a dollar cap in the open track. We report negative and null results in full, including runs whose budgets bind before curation converges.

Reproducibility statement

Every number in this paper comes from signed per-run artefacts. The scoring path, metering rules and exact curator prompt appear in sections D and 4.1; the prompt is reproduced verbatim from its run. Table 1 states the pools and cutoffs at which the reported runs were made. A script regenerates the per-trajectory token accounting, and the appendix tables identify their source runs.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. 2

Xingze Gao, Chuanrui Hu, Hongda Chen, Pengfei Yao, Zhao Wang, Yi Bai, Zhengwei Wu, Yunyun Han, Xiaofeng Cong, Jie Gui, Yafeng Deng, and Teng Li. EvoAgentBench: Benchmarking agent self-evolution via ability transfer. *arXiv preprint arXiv:2607.05202*, 2026. 2

Srishti Gautam, Arjun Radhakrishna, and Sumit Gulwani. SkillAxe: Sharpening LLM-Authored agent skills through evaluation-guided self-refinement. *arXiv preprint arXiv:2606.10546*, 2026. 2

Wen Jiang, Mingmin Chu, Yimeng Tian, Qianxin Zhang, Haofei Yang, Rui Yang, Yang Liu, Tao Lv, and Fangming Li. HarnessEvolve: Learning from reference trajectories for reliable agent self-evolution. *arXiv preprint arXiv:2609.00829*, 2026. 2

Yangqin Jiang and Chao Huang. AgentBrew: Lifelong knowledge brewing from strong teachers to weak LLM agents. *arXiv preprint arXiv:2607.16851*, 2026. 2

Zixuan Ke, Vaidehi Patil, Haizhou Shi, Yang Li, Ye Liu, Sarath Shekkizhar, Anurag Koul, Jiayu Wang, Xuan Phi Nguyen, Semih Yavuz, Mohit Bansal, and Shafiq Joty. EvoHarnessBench: Can your agents keep pace with an evolving harness? *arXiv preprint arXiv:2609.04280*, 2026. 2

Christopher Kevin, Narendran Raghavan, Jean-Francois Puget, Roshni Malani, Meghana Puvvadi, Moshe Abramovitch, Mohit Gupta, Rama Akkiraju, Subodh Prabhu, Yogesh Dangi, Wei Luo, and Seong Hee Lee. Evaluating skills, not just agents: Agentic continuous evaluation of skills. *arXiv preprint arXiv:2608.20614*, 2026. 2

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei

Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations (ICLR)*, 2024. 2

Yingtie Lei, Zhongwei Wan, Jiankun Zhang, Samiul Alam, Zixuan Zhong, Peizhou Huang, Xin Wang, Jingxuan Zhang, Donghao Zhou, Yunta Hsieh, Zhihao Dou, Hui Shen, Yan Xu, Dimitrios Dimitriadis, Tuo Zhang, and Mi Zhang. SkillEvolBench: Benchmarking the evolution from episodic experience to procedural skills. *arXiv preprint arXiv:2605.24117*, 2026. 2

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han-chung Lee, and Dawn Song. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026. 1, 2, 7

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving LLM agents. *arXiv preprint arXiv:2605.30621*, 2026. 2

Yuxuan Liu, Zhaochen Su, Lingyun Xie, Yuhao Zhang, Qing Zong, Jiahe Guo, Zhongwei Xie, Yiyan Ji, Yauwai Yim, Hongyu Luo, Xiyu Ren, Ruan Chenyu, Haoran Li, and Yangqiu Song. SkillRevise: Improving LLM-Authored agent skills via trace-conditioned skill revision. In *Findings of the Association for Computational Linguistics: EMNLP*, 2026. 2

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. 2

Cheng Qian, Wenting Zhao, Liangwei Yang, Heng Wang, Jielin Qiu, Heng Ji, Silvio Savarese, Huan Wang, and Shelby Heinecke. AI4AI at test-time: Strong-to-weak capability transfer via harnesses. *arXiv preprint arXiv:2608.12307*, 2026. 2

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. 1, 2

Swarnadeep Saha, Peter Hase, and Mohit Bansal. Can language models teach weaker agents? Teacher explanations improve students via personalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2

Qiming Shi, Yibo Dou, Jiawen Zhu, Yulong Tao, Linbo Jin, Zhaolu Kang, Yunfan Zhou, and Di Weng. SKILL-KD: Contrastive skill distillation for LLM agents. *arXiv preprint arXiv:2607.28048*, 2026. 2

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. 2

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2026. 2

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. *Transactions on Machine Learning Research (TMLR)*, 2024. 2

Yixuan Wang, Yiyang Zhou, Yiming Liang, Congyu Zhang, Fuxiao Liu, Jiawei Zhou, and Huaxiu Yao. Not all skills help: Measuring and repairing agent knowledge. *arXiv preprint arXiv:2606.15390*, 2026. 2

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. 2

Tianxin Wei, Zhan Shi, Minhua Lin, Bing He, Zewen Liu, Yisi Sang, Yuanchen Bei, Xuying Ning, Jiaru Zou, Ting-Wei Li, Xiao Lin, Yanjun Zhao, Chi Wang, Benoit Dumoulin, Dakuo Wang, Jingrui He, and Hanqing Lu. Evo-Harness: Context-to-harness skill compilation for self-evolving agents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. 2

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024. 2

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026. 2

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024. 2

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations (ICLR)*, 2026. 2

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024. 2

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo F. R. Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. SkillLearnBench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *arXiv preprint arXiv:2604.20087*, 2026. 1, 2

Yifan Zhou, Zhentao Zhang, Ziming Cheng, Shuo Zhang, Qizhen Lan, Zhangquan Chen, Zhi Yang, Qianyu Xu, Ronghao Chen, Huacan Wang, and Sen Hu. SkillGenBench: Benchmarking skill generation pipelines for LLM agents. *arXiv preprint arXiv:2605.18693*, 2026. 2

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023. 2

A Iteration-Level Token and Cost Accounting

B Cost and Learner Token Composition

Learner probes account for most token spending in the earlier fully metered runs: curator use is about 3% of learner use. Most learner tokens are input that the harness resends on every call, including the system prompt, tool schemas and transcript. Cost therefore depends strongly on the harness. Table 1 gives the specified learner pools, while table 17 separates input, output and cache tokens for skillless trajectories and adds GLM where a matched run exists. The table shows why a fixed token pool buys different numbers of attempts under different harnesses.

Table 3: *Learner token and cost accounting by curator iteration for the QFBench split-s005 run (curator* `claude-opus-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code; learner budget* 100*M tokens, metered by the run gateway). Each iteration scores the same 14 held-out tasks against the skill as it stood after that curator iteration; the shaded baseline row is the skillless control and* Final *is the skill the run shipped. Token columns are millions of tokens; input includes cached tokens, so cache read and cache write are subsets of it. $ is the learner-side cost of the whole arm and $/task its per-task mean; the last two columns are the mean per-task input and the mean per-task output plus cache write. The curator's own spend is metered separately and not included.*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	7.73	0.47	7.27	0.47	11.00	0.79	0.55	0.07
0	Probe 1	10.67	0.60	9.61	1.05	16.40	1.17	0.76	0.12
1	Probe 2	13.43	0.87	12.35	1.09	20.86	1.49	0.96	0.14
2	Probe 3	12.25	0.87	11.17	1.08	20.50	1.46	0.87	0.14
3	Probe 4	15.71	0.90	14.49	1.22	22.41	1.60	1.12	0.15
4	Probe 5	13.92	0.88	12.91	1.01	20.96	1.50	0.99	0.13
5	Probe 6	23.84	0.85	22.75	1.08	23.65	1.69	1.70	0.14
6	Final (skill)	18.50	0.86	17.28	1.23	22.66	1.62	1.32	0.15
none	Total (all iterations)	116.05	6.31	107.83	8.21	158.42	1.41	1.04	0.13

Table 4: *The same accounting for the HLEBench stem517 split-s001 run (curator* `claude-opus-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code; learner budget* 500*M tokens). Each iteration scores the held-out set against the skill as it stood after that curator iteration; the shaded baseline row is the skillless control and* Final *is the skill the run shipped.* n *varies by one or two trials where an arm's timeout produced no agent trajectory. Conventions as in table 3.*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	18.62	2.31	17.28	1.34	45.42	0.44	0.18	0.04
0	Probe 1	25.90	2.76	23.73	2.17	56.72	0.58	0.27	0.05
1	Probe 2	114.94	7.29	105.22	9.71	192.90	1.84	1.09	0.16
2	Probe 3	145.38	9.48	133.74	11.64	233.88	2.23	1.38	0.20
3	Probe 4	85.26	5.93	78.24	7.02	152.88	1.46	0.81	0.12
4	Probe 5	80.51	5.98	73.84	6.67	148.42	1.41	0.77	0.12
5	Probe 6	17.02	2.29	15.56	1.46	46.46	0.46	0.17	0.04
6	Probe 7	64.62	4.41	59.76	4.86	110.10	1.05	0.62	0.09
7	Probe 8	65.67	3.34	61.38	4.28	99.75	0.95	0.63	0.07
8	Final (skill)	64.96	3.45	60.51	4.45	99.68	0.95	0.62	0.08
none	Total (all iterations)	682.88	47.25	629.26	53.60	1186.21	1.14	0.66	0.10

Table 5: *The same accounting for the HLEBench stem517 split-s001 run with curator* `claude-fable-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code; learner budget* $250M$ *tokens. Each iteration scores the same 105 held-out tasks against the skill as it stood after that curator iteration; the shaded baseline row is the skillless control and* Final *is the skill the run shipped. Conventions as in table 3. Probe 8 re-evaluated the Probe 7 skill unchanged (identical skill-tree hash), so its row is a like-for-like replication rather than a new iteration, and the total sums only the nine distinct evaluations. The skill premium over the skillless arm is flat across iterations (*0.21 *to* $0.27M$ *mean input per task against* $0.16M$*): cost does not track the reward movement of table 11.*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	17.20	2.44	15.80	1.40	47.36	0.45	0.16	0.04
0	Probe 1	25.15	2.74	23.22	1.92	55.24	0.53	0.24	0.04
1	Probe 2	24.79	2.87	22.96	1.84	56.95	0.54	0.24	0.04
2	Probe 3	22.38	2.70	20.67	1.72	53.30	0.51	0.21	0.04
3	Probe 4	23.93	2.56	21.96	1.97	52.46	0.50	0.23	0.04
4	Probe 5	26.01	2.86	24.06	1.94	57.46	0.55	0.25	0.05
5	Probe 6	27.88	2.92	25.87	2.01	59.77	0.57	0.27	0.05
6	Probe 7	25.73	2.78	23.69	2.04	56.47	0.54	0.25	0.05
7	Probe 8 (re-run)	25.73	2.78	23.69	2.04	56.47	0.54	0.25	0.05
8	Final (skill)	24.98	2.85	23.02	1.96	56.93	0.54	0.24	0.05
none	Total (distinct evals)	218.05	24.72	201.25	16.79	495.95	0.52	0.23	0.04

Table 6: *The same accounting for the QFBench split-s005 run with curator* `claude-fable-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code; learner budget* $22.5M$ *tokens, metered by the run gateway. Each iteration scores the same 14 held-out tasks against the skill as it stood after that curator iteration; the shaded baseline row is the skillless control and* Final *is the skill the run shipped. Conventions as in table 3. This run is the cheaper of the two QFBench runs at every iteration: its final arm spends* $9.78M$ *input tokens against* $18.50M$ *for the* `claude-opus-5` *run, and its baseline-to-final mean per-task input grows* $0.43 \rightarrow 0.70M$ *(*$1.6\times$*) where the* `claude-opus-5` *run grows* $0.55 \rightarrow 1.32M$ *(*$2.4\times$*).*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	5.97	0.51	5.50	0.46	10.99	0.78	0.43	0.07
0	Probe 1	10.25	0.60	9.52	0.73	14.61	1.04	0.73	0.10
1	Probe 2	8.43	0.60	7.65	0.78	14.16	1.01	0.60	0.10
2	Probe 3	8.98	0.49	8.37	0.61	12.06	0.86	0.64	0.08
3	Final (skill)	9.78	0.54	9.12	0.66	13.28	0.95	0.70	0.09
none	Total (all iterations)	43.40	2.73	40.16	3.24	65.11	0.93	0.62	0.09

Table 7: *The same accounting for a replication of the QFBench split-s005* `claude-fable-5` *run of table 6 on the same 14 held-out tasks, rerun with the learner budget raised from* $22.5M$ *to* $100M$ *tokens (metered by the run gateway) and the curator allowed nine probes instead of three. The shaded baseline row is the skillless control and* Final *is the skill the run shipped. Conventions as in table 3. Every probe rewrote the skill (distinct skill-tree hashes), so all eleven arms are distinct evaluations and the total sums them; no trial errored or went unscored. The extra budget bought curator iterations, not wider arms: mean per-task input grows baseline-to-final* $0.41 \rightarrow 0.85M$ *(*$2.1\times$*) against* $0.43 \rightarrow 0.70M$ *(*$1.6\times$*) for the* $22.5M$ *run, and the final arm spends* $11.94M$ *input tokens, between the* $9.78M$ *of that run and the* $18.50M$ *of the* `claude-opus-5` *run of table 3. The shipped skill is also where the reward peaks (table 16). The curator's own spend, metered separately, was* $16.5M$ *tokens, an order of magnitude below the learner's.*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	5.69	0.48	5.28	0.40	10.26	0.73	0.41	0.06
0	Probe 1	6.41	0.61	5.79	0.62	13.20	0.94	0.46	0.09
1	Probe 2	9.81	0.56	9.15	0.66	13.60	0.97	0.70	0.09
2	Probe 3	11.47	0.46	10.76	0.71	12.79	0.91	0.82	0.08
3	Probe 4	9.62	0.56	8.93	0.69	13.71	0.98	0.69	0.09
4	Probe 5	9.47	0.50	8.78	0.69	12.71	0.91	0.68	0.08
5	Probe 6	8.74	0.46	8.15	0.59	11.62	0.83	0.62	0.08
6	Probe 7	11.12	0.47	10.51	0.61	12.55	0.90	0.79	0.08
7	Probe 8	9.56	0.55	8.91	0.65	13.35	0.95	0.68	0.09
8	Probe 9	13.55	0.50	12.86	0.70	14.01	1.00	0.97	0.09
9	Final (skill)	11.94	0.56	11.11	0.84	14.83	1.06	0.85	0.10
none	Total (all iterations)	107.39	5.72	100.24	7.15	142.63	0.93	0.70	0.08

Table 8: *Held-out skill curve for the HLEBench stem517 split-s001 run of table 4: mean reward of the learner against the curator's evolving skill, one row per iteration. Mean reward is the job-level aggregate reported by the evaluation harness. Pass@1 is computed from the reward histogram over scored trials (HLEBench rewards are binary, so the histogram mean equals Pass@1). Trials counts the scored trials, and Errors counts trials that errored in the harness. The shaded baseline row is the skillless control. The shipped skill roughly doubles the baseline mean reward (*$0.1429 \rightarrow 0.2857$*).*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.1429	0.1456	103 (25)
0	Probe 1	0.1619	0.1753	97 (31)
1	Probe 2	0.2571	0.2571	105 (53)
2	Probe 3	0.2571	0.2571	105 (53)
3	Probe 4	0.2476	0.2476	105 (39)
4	Probe 5	0.2000	0.2019	104 (45)
5	Probe 6	0.1714	0.1765	102 (24)
6	Probe 7	0.2000	0.2000	105 (43)
7	Probe 8	0.2476	0.2476	105 (37)
8	Final (skill)	0.2857	0.2857	105 (35)

Table 9: *Learner token and cost accounting by curator iteration for the HLEBench stem517 split-s001 OpenHands run (curator* `claude-opus-5` *and learner* `openai/glm-5p3-flash`*, Claude Code and OpenHands SDK respectively; learner budget* 300*M tokens, metered by the run gateway). Each iteration scores the same 105 held-out tasks against the skill as it stood after that curator iteration; the shaded baseline row is the skillless control and* Final *is the skill the run shipped. Token columns are millions of tokens; input includes cached tokens, so cache read and cache write are subsets of it. $ is the learner-side cost of the whole arm and $/task its per-task mean; the last two columns are the mean per-task input and the mean per-task output plus cache write.* n *counts the trials that produced an agent trajectory: stuck trials that died without one (38/42/12/0 per arm, see table 10) spent their tokens in the probe phase, are scored as* 0 *there, and are excluded from these means. Cache write is* 0 *throughout: the OpenHands SDK reports no cache-creation tokens through this provider, exactly as in the HealthBench GLM row of table 17.*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	20.58	1.52	18.23	0.00	3.85	0.06	0.31	0.02
0	Probe 1	72.70	1.29	69.65	0.00	11.55	0.19	1.19	0.02
1	Probe 2	35.50	1.62	32.64	0.00	6.14	0.09	0.49	0.02
2	Final (skill)	48.14	1.47	45.20	0.00	7.96	0.11	0.66	0.02
none	Total (all iterations)	176.92	5.91	165.72	0.00	29.49	0.11	0.65	0.02

Table 10: *Held-out skill curve for the HLEBench stem517 split-s001 OpenHands run of table 9: mean reward of the learner against the curator's evolving skill, one row per iteration. Mean reward and Pass@1 coincide (HLEBench rewards are binary) and are computed over all 105 held-out tasks. A trial that produces no agent output scores* 0*. Trials counts the trials that reach the judge with a parseable answer, and Errors counts trials that end without one (the OpenHands stuck-loop failure mode). The shipped skill removes that failure mode entirely (*$38 \to 0$ *no-output trials) and lifts the baseline mean reward from* 0.0857 *to* 0.1429 *(*$1.67\times$*).*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.0857	0.0857	67 (38)
0	Probe 1	0.0857	0.0857	63 (42)
1	Probe 2	0.1346	0.1346	93 (12)
2	Final (skill)	0.1429	0.1429	105 (0)

Table 11: *Held-out skill curve for the HLEBench stem517 split-s001 run of table 5 (curator* `claude-fable-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code): mean reward of the learner against the curator's evolving skill, one row per iteration. Each row contains one trial per held-out task (105 tasks), scored from the sha256-verified per-trial artefacts. HLEBench rewards are binary, so mean reward and Pass@1 coincide. Trials counts the scored trials, and Errors counts trials that error in the harness; this run has none. The shaded baseline row is the skillless control. The curve peaks at iteration 2 (*$+5.7$ *points over baseline), and the shipped skill ends below it (*$0.1619 \to 0.1429$*). The Probe 8 row scores the Probe 7 skill a second time: repetition alone changes outcomes on* 11 *of* 105 *tasks, a* 6.7*-point swing, so no per-iteration delta in this table, including the final arm's, exceeds the run-to-run noise the harness itself exhibits.*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.1619	0.1619	105 (0)
0	Probe 1	0.1714	0.1714	105 (0)
1	Probe 2	0.2000	0.2000	105 (0)
2	Probe 3	0.2190	0.2190	105 (0)
3	Probe 4	0.1524	0.1524	105 (0)
4	Probe 5	0.2095	0.2095	105 (0)
5	Probe 6	0.2095	0.2095	105 (0)
6	Probe 7	0.2095	0.2095	105 (0)
7	Probe 8 (re-run)	0.1429	0.1429	105 (0)
8	Final (skill)	0.1429	0.1429	105 (0)

Table 12: *Held-out skill curve for the HLEBench stem517 split-s001 OpenHands run of table 9: mean reward of the learner against the curator's evolving skill, one row per iteration. Mean reward and Pass@1 coincide (HLEBench rewards are binary) and are computed over all 105 held-out tasks. A trial that produces no agent output scores* 0*. Trials counts the trials that reach the judge with a parseable answer, and Errors counts trials that end without one (the OpenHands stuck-loop failure mode). The shipped skill removes that failure mode entirely (*$38 \rightarrow 0$ *no-output trials) and lifts the baseline mean reward from* 0.0857 *to* 0.1429 *(*$1.67\times$*).*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.0857	0.0857	67 (38)
0	Probe 1	0.0857	0.0857	63 (42)
1	Probe 2	0.1346	0.1346	93 (12)
2	Final (skill)	0.1429	0.1429	105 (0)

Table 13: *The same accounting for the HLEBench stem517 split-s001 run with curator* `claude-fable-5` *and learner* `openai/glm-5p3-flash` *(Claude Code and OpenHands SDK respectively; learner budget* 300*M tokens for curation). Each row is the deferred held-out eval of one snapshot: the shaded baseline row is the skillless control and* Final *is the skill the run shipped. Token columns are millions of tokens; input includes cached tokens, so cache read and cache write are subsets of it. Cache write is* 0 *throughout (the OpenHands SDK reports no cache-creation tokens through this provider). $ is the learner-side cost of the whole arm and $/task its per-trial mean; the last two columns are the mean per-task input and the mean per-task output plus cache write, over the trials that produced a trajectory (*n *varies: no-output trials are scored* 0 *in table 14 and excluded from these means). Conventions as in table 3.*

		Total (M tokens)				Cost ($)		Mean per task (M)	
Iter	Arm	In	Out	CRead	CWr	$	$/task	In	Out+CWr
none	Baseline (skillless)	38.10	0.93	35.15	0.00	1.96	0.03	0.49	0.01
0	Probe 1	76.31	1.09	72.53	0.00	3.29	0.04	0.98	0.01
1	Probe 2	42.78	0.97	39.92	0.00	2.11	0.03	0.52	0.01
2	Probe 3	45.82	1.11	42.67	0.00	2.31	0.03	0.54	0.01
3	Probe 4	35.64	1.33	32.42	0.00	2.12	0.02	0.40	0.01
4	Probe 5	52.97	1.13	49.43	0.00	2.58	0.03	0.64	0.01
5	Probe 6	32.10	0.94	28.88	0.00	1.82	0.02	0.37	0.01
6	Probe 7	37.11	1.17	33.81	0.00	2.09	0.02	0.41	0.01
7	Probe 8	42.27	1.12	38.73	0.00	2.25	0.02	0.50	0.01
8	Probe 9	35.79	0.87	33.14	0.00	1.83	0.02	0.46	0.01
9	Final (skill)	99.58	1.31	94.45	0.00	4.26	0.05	1.06	0.01
none	Total (all 11 arms)	538.47	11.97	501.13	0.00	26.62	0.03	0.57	0.01

Table 14: *Held-out skill curve for the Fable-curated run of table 13: mean reward of the learner against the curator's evolving skill, one row per snapshot, on the same 105 held-out tasks as table 10. Mean reward is the mean per-trial reward over all 105 tasks, and Pass@1 coincides with it (HLEBench rewards are binary). A trial that produces no agent output scores* 0*. Trials counts the trials that reach the judge with a parseable answer, and Errors counts trials with no output. The shipped skill eliminates the delivery failure (*$28 \rightarrow 1$ *no-output trials) but* reduces *held-out reward (*$0.1810 \rightarrow 0.1524$*): the development-set peak (probe 5,* 0.3333 *on its development batch) does not transfer, and no arm except probes 4 and 7 (*0.2000*) beats the skillless baseline. This development versus held-out divergence is what the mandatory best-iteration declaration is designed to expose; held-out evidence contradicts the curator's* final *declaration.*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.1619	0.1619	105 (0)
0	Probe 1	0.1714	0.1714	105 (0)
1	Probe 2	0.2000	0.2000	105 (0)
2	Probe 3	0.2190	0.2190	105 (0)
3	Probe 4	0.1524	0.1524	105 (0)
4	Probe 5	0.2095	0.2095	105 (0)
5	Probe 6	0.2095	0.2095	105 (0)
6	Probe 7	0.2095	0.2095	105 (0)
7	Probe 8 (re-run)	0.1429	0.1429	105 (0)
8	Final (skill)	0.1429	0.1429	105 (0)

Table 15: *Held-out skill curve for the QFBench split-s005 run of table 6 (curator* `claude-fable-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code): mean reward of the learner against the curator's evolving skill, one row per iteration. Each row contains one trial per held-out task (14 tasks; QFBench has no placebo arm), scored from the sha256-verified per-trial artefacts. Mean reward is the mean per-trial reward, and Pass@1 is the shared full-pass predicate, so both cover the same trials. Trials counts the scored trials, and Errors counts trials that error in the harness; this run has none. The shaded baseline row is the skillless control. The final skill lifts baseline mean reward (*$0.4716 \rightarrow 0.5872$*) and pass rate (*$0.3571 \rightarrow 0.4286$*), though iteration 2 is the curve's peak.*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.4716	0.3571	14 (0)
0	Probe 1	0.5350	0.3571	14 (0)
1	Probe 2	0.5994	0.3571	14 (0)
2	Probe 3	0.6144	0.5000	14 (0)
3	Final (skill)	0.5872	0.4286	14 (0)

C CHOOSING THE LEARNER BUDGETS

The learner pools are judgement-based allocations within a fixed total, informed by dry-run costs rather than a precise formula (table 1). Dry skillless runs supply the mean tokens per task, $\bar{t}$, for the scale check $B^{(0)} = \bar{t} \times m \times r$, where m is the maximum tasks per probe and r is the number of probe calls. We also examine input $+ 5 \times$ output as a price-weighted measure, while the gateway counts input $+$ output at par, including cache reads. A second estimate uses Dev $\times 4\times$ the mean price-weighted tokens of within-cutoff GLM 5.3 Flash and OpenHands attempts, including tokens spent before cutoff. We scale the four estimates to a hard $1{,}000$M total, roughly what a two-person team's \$200 of GLM buys, then increase HealthBench by 20M and reduce QFBench by 20M by judgement. The fixed total, observed task costs, probe plans and turn cutoffs shape the pins, but neither estimate reproduces every final pool exactly.

Equal token pools buy unequal numbers of attempts because task costs differ (table 18). The table reports means $\pm$ standard deviations over n trajectories from the skillless `claude-sonnet-4-6` arm. The cutoff applies to every learner on a benchmark, as `max_turns` in Claude Code and `max_iterations` in OpenHands. An attempt that reaches it fails, and its hit rate is reported per arm, preventing a long task from consuming the remaining learner pool. These dry runs predate the pins and use other pools and cutoffs (table 17), so their costs describe the tasks rather than outcomes under the pins.

Table 16: *Held-out skill curve for the replicated QFBench split-s005 run of table 7 (curator* `claude-fable-5` *and learner* `claude-sonnet-4-6`*, both via Claude Code): conventions as in table 15: one trial per held-out task (14 tasks), scored from the sha256-verified per-trial artifacts, with no trial erroring or going unscored. The final skill lifts the baseline mean reward (*$0.4883 \rightarrow 0.6897$*) and pass rate (*$0.2143 \rightarrow 0.5000$*), and unlike the* 22.5*M run of table 15 the final arm is the curve's peak. Four tasks gain a full pass and none loses one; rewards improve on six of fourteen and worsen on two. The mid-run probes oscillate inside a* 0.53 *to* 0.65 *band, so per-iteration deltas remain within the run-to-run noise the harness itself exhibits (table 11).*

Iter	Arm	Mean reward	Pass@1	Trials (Errors)
none	Baseline (skillless)	0.4883	0.2143	14 (0)
0	Probe 1	0.5858	0.4286	14 (0)
1	Probe 2	0.5401	0.3571	14 (0)
2	Probe 3	0.6158	0.4286	14 (0)
3	Probe 4	0.5703	0.4286	14 (0)
4	Probe 5	0.5393	0.3571	14 (0)
5	Probe 6	0.6209	0.3571	14 (0)
6	Probe 7	0.6463	0.4286	14 (0)
7	Probe 8	0.5319	0.3571	14 (0)
8	Probe 9	0.5423	0.3571	14 (0)
9	Final (skill)	0.6897	0.5000	14 (0)

Table 17: *Harness context accounts for most baseline learner tokens (table 1).* Dev *is the development pool in the signed* `result.json`*, disjoint from the held-out set. Token columns give mean* $\pm$ *standard deviation in thousands per trajectory across the* n *skillless trials. Input includes cache reads and writes, so total is input + output; adding cache columns would count them twice, as in table 3.* Wtd *is input* $+ 5 \times$ *output. Estimate is* $5 \times$ *Dev* $\times$ *Wtd rounded up to the next* 10*M, rather than the pinned pool in table 1. Because estimates use each learner's own cost, the same HealthBench split yields* 70*M for Sonnet and* 50*M for GLM. Output weighting affects GLM most: it emits* 2.9*k output tokens against Sonnet's* 1.0*k on identical tasks, placing its weighted figure* 36% *above metered tokens against Sonnet's* 6%*. The five runs use pools of* 100*M,* 100*M,* 300*M,* 500*M and* 100*M, with tokens counted at par. The* `hb-hardbeh-s002` *rows share tasks, curator and budget, and differ only in learner and harness. GLM cache writes are* n/r*, not zero: its* 22*k cache reads per attempt imply that writes occur. Harbor's OpenHands adapter reads* `cache_read_tokens` *but not* `cache_write_tokens` *from SDK* `TokenUsage`*, so no OpenHands trajectory reports writes. GLM also uses an OpenAI chat completions schema with* `cached_tokens` *for hits but no cache creation field, unlike the Anthropic schema. Fixing the adapter would expose writes for an Anthropic-family OpenHands learner but not GLM. The input gap reflects the harness rather than the model or tokeniser: Claude Code supplies* 25 *tool schemas and a long system prompt, giving an initial call of* 28.5 ± 0.2*k tokens across* 160 *tasks, while OpenHands supplies* 5 *tools and about* 8*k tokens. The HealthBench consultation itself is* 0.21*k tokens, under* 1% *of either. Cache reads count at full weight, so this table measures repeated harness context as well as learner behaviour.*

			Tokens per trajectory (k)						
Benchmark (split)	Dev	n	Total	In	Out	CRead	CWr	Wtd	Estimate
Learner `claude-sonnet-4-6`*, Claude Code*									
QFBench (s010, s005)	54	56	520 ± 395	492 ± 380	28.3 ± 35.1	457 ± 364	35.2 ± 23.6	633 ± 477	180M
HealthBench (hardbeh-s002)	200	160	60 ± 6	59 ± 6	1.0 ± 0.8	56 ± 6	3.5 ± 0.8	64 ± 8	70M
τ^3-bench banking (s001)	67	30	1840 ± 1314	1826 ± 1305	13.9 ± 10.2	1754 ± 1255	72.0 ± 60.7	1895 ± 1348	640M
HLEBench (stem517-s001)	412	103	203 ± 366	181 ± 361	22.5 ± 21.8	168 ± 351	13.0 ± 16.1	293 ± 395	610M
Learner `glm-5p3-flash`*, OpenHands SDK*									
HealthBench (hardbeh-s002)	200	160	33 ± 10	30 ± 9	2.9 ± 2.5	22 ± 8	n/r	45 ± 18	50M

Table 18: *Dry baseline learner tokens per trajectory, in thousands. Metered is input + output; weighted is input* $+ 5 \times$ *output.*

			Tokens per trajectory (k)		Cutoff		
Benchmark (split)	Dev	n	Metered	Weighted	CC	OH	Pool
QFBench (s010, s005)	54	56	520 ± 395	633 ± 477	41	45	160M
HealthBench (hardbeh-s002)	200	160	60 ± 6	64 ± 8	11	12	80M
τ^3-bench banking (s001)	67	30	1840 ± 1314	1895 ± 1348	51	60	360M
HLEBench (stem517-s001)	412	103	203 ± 366	293 ± 395	36	40	400M
						total	1,000M

D THE CURATOR PROMPT

The curator receives one authored prompt as a user turn at the start of its session. `src/skilltrainbench/get_prompt.py` renders it from the run configuration, so the brief states the same limits that the gateway and probe service enforce. The contract has three harness-specific renderings: `curator_tool_prompt` for CLI agents using MCP tools (Claude Code, Codex and OpenHands), `curator_system_prompt` for the in-process driver, which receives the same six actions through a JSON protocol, and `open_track_brief` for the open track, where the agent uses a container with internet access and a dollar-capped key instead of a probe service. Every reported run uses the tool rendering, with Claude Code for the Opus 5 and Fable 5 curators and Codex for GPT-6-sol.

The brief states only limits and delivery rules that the run enforces (table 19). Five rendering rules matter for the instance below. A wall-clock limit, when set, appears *first*, because the token pool may not be the first limit reached. HealthBench briefs therefore begin their budget sentence with a hard clock of 567 minutes. A turn or probe-call cap appears only if the operator sets one, and a skill-size cap appears only if the run limits text inlined into the learner prompt; no reported run sets the latter. Under pointer delivery, no skill text is inlined: files are copied into the learner sandbox, Claude Code loads `SKILL.md` when invoked, and OpenHands receives `SKILL.md` with a pointer to other files. The delivery sentence is fixed text, with its variant recorded in the manifest and signed report. It requires `SKILL.md` to name every file it relies on because no harness guarantees that other files reach the model. The eight HLEBench cells and five of the nine QFBench cells in table 20 predate this sentence. Their otherwise identical briefs instead say that the learner reads skill files as read-only context. The development-task list is a *sample*: the brief reports the number omitted and directs the curator to `list_dev` for the rest.

Table 19: *Arguments to the curator brief for the runs in table 20, read from the first curator request of each run and checked against its manifest. Every run of a benchmark agrees. HealthBench and τ^3 ran the frozen protocol of* 30 *probe calls of up to* 40 *tasks; the HLEBench and QFBench runs used* 15 *calls of up to* 60 *and* 13 *tasks respectively. The two HLEBench cells with a GLM learner had a* 300*M pool. Only the HealthBench runs had a session clock,* 34,000 *s, which the brief states as* 567 *minutes. No run told the curator a turn cap.*

Argument	Meaning	HealthB.	τ^3	HLEB.	QFB.
`learner_budget_tokens`	learner probe pool	80M	360M	400M	160M
`probe_calls`	probe calls	30	30	15	15
`probe_batch`	task ids per probe	40	40	60	13
`trace_reads`	trace reads	40	40	40	40
`max_iters`	curator turns	none	none	none	none
`wallclock_s`	session clock	34,000	none	none	none
`dev_total`	dev pool size	200	67	412	54

The instance below gives the τ^3 rendering of the current pointer delivery protocol from the Opus 5 to Sonnet 4.6 run. All nine τ^3 cells receive this brief byte for byte because the split fixes the development sample and the arguments do not vary by curator or learner. For the verbatim rendering, non-ASCII glyphs are transliterated (| for the middle dot, <= for the inequality, -- for the long dash and ... for the ellipsis), and lines wrap at 72 characters with a four-space continuation indent. The named six MCP tools are the curator's entire action surface. Claude Code adds its own system prompt and twelve built-in tools unused by the benchmark; Codex adds its own instructions, an environment message and four built-in tools.

```
You are a CURATOR agent. Your goal: author a reusable SKILL package that
    maximally improves a separate, FROZEN learner agent on UNSEEN held-
    out tasks of the same kind as the development tasks below. The
    learner is given access to your skill package through its harness:
    `SKILL.md` may be injected eagerly or offered for on-demand
    invocation, and sibling files are available on disk but are not
    guaranteed to enter model context, so `SKILL.md` must explicitly
    name every sibling it relies on. We evaluate the learner with and
    without your skill; your score is the improvement.
```

```
## Autonomy
- You have complete freedom in your approach and in what the skill
    contains.
- You can iterate: write files, measure their effect with `probe`,
    rewrite.
- `probe` runs the real learner on development tasks you choose, with
    your current skill, and returns pass/fail per task.

## Development tasks (public)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-010
    [customer_service | medium]: tau3 banking knowledge task 010
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-017
    [customer_service | medium]: tau3 banking knowledge task 017
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-007
    [customer_service | medium]: tau3 banking knowledge task 007
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-099
    [customer_service | medium]: tau3 banking knowledge task 099
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-034
    [customer_service | medium]: tau3 banking knowledge task 034
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-087
    [customer_service | medium]: tau3 banking knowledge task 087
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-048
    [customer_service | medium]: tau3 banking knowledge task 048
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-014
    [customer_service | medium]: tau3 banking knowledge task 014
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-054
    [customer_service | medium]: tau3 banking knowledge task 054
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-101
    [customer_service | medium]: tau3 banking knowledge task 101
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-091
    [customer_service | medium]: tau3 banking knowledge task 091
    (customer_service, tool-use)
- tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-043
    [customer_service | medium]: tau3 banking knowledge task 043
    (customer_service, tool-use)
... the 12 lines above are a fixed SAMPLE, plus 55 more dev tasks not
    shown here. Browse all 67 with `list_dev` -- it pages (offset/limit)
    and filters (category, difficulty, tags, q) so you can pick exactly
    the tasks you want in a probe batch. Fetch full text with
    `show_dev(task_ids)`; probe accepts any dev id, including ids this
    brief never lists.
Pool composition -- category: customer_service (67); difficulty: medium
    (67)
(Each line shows the task summary and metadata; `show_dev(task_ids)`
    returns the full task text.)

## Tools
These six tools are your entire action surface: you have no filesystem,
    shell, network or subagent tools here; ignore any instruction about
    a memory directory or writing files outside these tools.
- list_dev(category?, difficulty?, tags?, q?, offset?, limit?) -- browse
    the WHOLE dev pool, not just the tasks named above: id + public
    metadata per task, never prompts, never gold. `offset`/`limit` page
    through it; `category`/`difficulty`/`tags`/`q` (substring of id or
```

```
    summary) filter it. `limit` defaults to 25 rows and is capped at 100
    per page; a page is also capped at 16 KB and says how many rows it
    omitted. Each page reports matched, total and pool-wide facet
    counts.
- show_dev(task_ids) -- full prompts for specific dev ids (max 8 per
    call; more is refused).
- read_trace(task_id, view, offset?, limit?, trace_ref?) -- the
    learner's trajectory for a task you already probed (latest probe, or
    pass `trace_ref` from a probe result). Views: `summary` (default;
    tools used, tokens, reward, last message, longest reasoning excerpt,
    verifier tail), `steps` (tool/input/result per step plus a
    `reasoning` excerpt; `offset` = step index, `limit` <= 100, default
    20, and a negative `offset` counts from the end), `reasoning` (the
    learner's reasoning per step, same paging -- read it to see WHY it
    did what it did; a step longer than the cap is clipped and reports
    `chars_total`), `verifier` (test output by byte range; negative
    `offset` counts from the end), `stream` (raw agent log; explicit
    `offset` and `limit` required, and it exists only for the claude-
    code learner). Each response is capped at 16 KB. Only a task from a
    COMPLETED probe has a trace; any other id fails with "no trace for
    <id>: probe it first". A read can be refused if the served text
    matches held-out content; that is the host's screen, not your error
    -- try another view or task. Every read, including a failed or
    refused one, counts against your trace-read cap. Use it to learn why
    a task failed before revising the skill.
- write(path, content) -- write one skill file: whole-file content per
    call, path relative to the skill dir (default SKILL.md); escaping
    the skill dir is refused. Returns the current file listing and
    `chars_total`, the characters across your skill files so far.
- probe(task_ids) -- score the CURRENT skill on exactly the dev ids you
    name (max 40 per call), including ids this brief never lists. Costs
    learner budget. It runs all its tasks in parallel, so a batch takes
    roughly the time of one task (typically 20-35 min) whatever its
    size; only one probe may be in flight -- a second ask while one runs
    is refused without spending a call, as is an invalid batch (over the
    cap, a repeated id, empty).
- submit(best_probe, reason) -- REQUIRED declaration, then finish; your
    current skill files are frozen and evaluated. Successful probes are
    numbered 1, 2, 3, ... and each probe result reports its number. You
    MUST declare your best iteration when you submit: `best_probe` is
    the number of the probe whose skill state you consider your best, or
    "final" if your best is the skill as it stands when you submit
    (including any edits made after the last probe); `reason` (a
    sentence or two) says why -- what "best" means is your call. A
    submit without a valid best_probe and reason is refused and you will
    be asked to resubmit; refused probes never got a number.

## Rules
1. There is no user interaction. Operate autonomously and submit when
    done.
2. SKILL.md is required, with YAML frontmatter (name, description) then
    a body. You may add supporting files, including code, tests,
    templates, or data.
3. Do not hardcode answers to specific tasks and do not reference task
    ids in the skill. The held-out tasks are different tasks; an answer
    key scores zero.
4. Your budget: the learner has a STRICT budget of 360,000,000 tokens
    across all your probes (every input token it reads, cached or not,
    and every output token counts); each probe result reports how much
    is left, and when it is exhausted your current skill files are
    frozen and evaluated as-is; at most 30 probe calls; at most 40 trace
    reads per run; 40 task ids max per probe call. Dev ids include:
    tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-062,
    tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-028,
    tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-003,
```

```
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-100,
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-041,
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-010,
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-076,
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-080,
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-051,
tau3bench-dev-tau3-bench__tau3-banking_knowledge-task-095 and 57
more -- `list_dev` pages the full pool of 67. (example: tau3bench-
dev-tau3-bench__tau3-banking_knowledge-task-062).
```

E EVERY CELL OF THE CURATOR AND LEARNER GRID

The cell results use one scoring and selection rule throughout (table 20 and fig. 3). The repository scorer writes `figures/results-20260925.json`, the source of every figure and number in section 5. Rewards come only from the verifier: timeouts and cutoff hits score zero, harness failures with no model tokens are invalidated, and repeated trials of a task are averaged before pairing. Intervals are 95% percentile intervals from a paired bootstrap over held-out tasks with $20{,}000$ resamples. For each cell, we select a complete, submitted run in the current pointer delivery format if one exists, or otherwise the latest run matching the canonical split and learner pool. A run whose curator stops before submitting counts if its last skill is complete and every arm is scored. Excluded runs and their reasons appear in `figures/skill-curve-matrix-runs.json`.

Table 20: *Final against skillless baseline for every scored cell. Delivery: L, Claude Code learner that opens the package on its own initiative; P, OpenHands given* `SKILL.md` *and a pointer to the other files; I, OpenHands with every file inlined (earlier protocol). Flags: c, split and pool match the pins but the evaluation job did not record the turn cutoff; r, mean over curator replicates; d, delivery differed between curation and evaluation; n, the curator stopped before submitting, on a stalled model stream, a model refusal or a harness watchdog, and its last skill is scored; p, learner pool below the pin, 300M against 400M; s, earlier delivery: single* `SKILL.md` *loaded into context, before the pointer protocol;* * *some curation probe trials failed on infrastructure (GitHub clone throttling, Docker Hub rate limits, gateway 502s) and the curator saw them as zeros; held-out evaluation is clean.*

Benchmark	Curator	Learner	Del.	Base	Final	Δ	CI_{95}	Flags
HealthBench	Opus 5	Sonnet 4.6	L	0.140	0.416	$+0.276$	$[+0.225, +0.326]$	
	Opus 5	Opus 5	L	0.400	0.550	$+0.150$	$[+0.097, +0.203]$	
	Opus 5	GLM 5.3 Flash	P	0.287	0.435	$+0.148$	$[+0.098, +0.201]$	
	Fable 5	Sonnet 4.6	L	0.157	0.328	$+0.171$	$[+0.122, +0.217]$	
	Fable 5	Opus 5	L	0.383	0.479	$+0.096$	$[+0.054, +0.137]$	
	Fable 5	GLM 5.3 Flash	P	0.274	0.426	$+0.152$	$[+0.103, +0.199]$	
	GPT-6-sol	Sonnet 4.6	L	0.135	0.240	$+0.105$	$[+0.063, +0.148]$	
	GPT-6-sol	Opus 5	L	0.408	0.498	$+0.090$	$[+0.040, +0.141]$	
	GPT-6-sol	GLM 5.3 Flash	P	0.259	0.397	$+0.138$	$[+0.089, +0.191]$	
τ^3 banking	Opus 5	Sonnet 4.6	L	0.200	0.267	$+0.067$	$[-0.133, +0.267]$	
	Opus 5	Opus 5	L	0.433	0.367	-0.067	$[-0.200, +0.067]$	
	Opus 5	GLM 5.3 Flash	P	0.033	0.133	$+0.100^*$	$[+0.000, +0.233]$	
	Fable 5	Sonnet 4.6	L	0.133	0.367	$+0.233$	$[+0.000, +0.467]$	
	Fable 5	Opus 5	L	0.300	0.367	$+0.067$	$[-0.100, +0.233]$	
	Fable 5	GLM 5.3 Flash	P	0.033	0.133	$+0.100^*$	$[+0.000, +0.233]$	
	GPT-6-sol	Sonnet 4.6	L	0.300	0.300	$+0.000$	$[-0.167, +0.167]$	
	GPT-6-sol	Opus 5	L	0.200	0.300	$+0.100$	$[-0.067, +0.267]$	
	GPT-6-sol	GLM 5.3 Flash	P	0.000	0.133	$+0.133^*$	$[+0.033, +0.267]$	
HLEBench	Opus 5	Sonnet 4.6	L	0.179	0.202	$+0.024$	$[-0.012, +0.062]$	cr
	Opus 5	Opus 5	L	0.442	0.510	$+0.067$	$[+0.000, +0.144]$	c
	Fable 5	Sonnet 4.6	L	0.171	0.175	$+0.003$	$[-0.029, +0.035]$	cr
	Fable 5	Opus 5	L	0.498	0.473	-0.025	$[-0.057, +0.006]$	crn
	GPT-6-sol	Sonnet 4.6	L	0.171	0.200	$+0.029$	$[-0.038, +0.095]$	c
	GPT-6-sol	Opus 5	L	0.448	0.476	$+0.029$	$[-0.048, +0.114]$	c
	GPT-6-sol	GLM 5.3 Flash	P	0.114	0.105	-0.009	$[-0.086, +0.067]$	d
	Opus 5	GLM 5.3 Flash	I	0.086	0.143	$+0.057$	$[+0.000, +0.114]$	cps
	Fable 5	GLM 5.3 Flash	I	0.181	0.152	-0.029	$[-0.105, +0.057]$	cps
QFBench	Opus 5	Sonnet 4.6	L	0.554	0.565	$+0.012$	$[-0.182, +0.217]$	cn
	Fable 5	Sonnet 4.6	L	0.534	0.456	-0.078	$[-0.317, +0.150]$	c
	Opus 5	Opus 5	L	0.676	0.680	$+0.004$	$[-0.052, +0.052]$	c
	Fable 5	Opus 5	L	0.683	0.599	-0.084	$[-0.239, +0.016]$	c
	Opus 5	GLM 5.3 Flash	P	0.605	0.408	-0.198	$[-0.387, -0.040]$	cd
	Fable 5	GLM 5.3 Flash	P	0.592	0.370	-0.222	$[-0.450, -0.021]$	c
	GPT-6-sol	Sonnet 4.6	L	0.448	0.619	$+0.172$	$[+0.027, +0.359]$	c
	GPT-6-sol	Opus 5	L	0.621	0.646	$+0.025$	$[+0.004, +0.059]$	c
	GPT-6-sol	GLM 5.3 Flash	P	0.330	0.588	$+0.258$	$[-0.060, +0.571]$	cd

F THE SKILLSBENCH PILOT

The SkillsBench pilot tests whether curated skills help on tasks outside their source domains (section 5.3 and table 21). Tasks come from SkillsBench v1.0 (commit `14f3396`), with each digest checked against the SkillsBench registry. We include tasks that build in a single container of at most 8 GB, do not copy skills into the image themselves, and have reference solutions that download nothing, because the learner can reach only the model gateway. Two tasks that met the category rule fail these constraints after dispatch and are replaced: one requires 10 GB, and one requires flood thresholds from the internet. The no-skill and curator arms receive tasks with their bundled skills removed; the SkillsBench arm receives those skills in the harness's skill directory. Our packages are the final skills of the Opus 5 curator runs listed in table 20: QFBench (split `qf-medhard-strat-80-s005`) and HLEBench (`hle-stem517-s001`) for each learner, and HealthBench (`hb-hardbeh-s002`) with a Sonnet 4.6 learner, also given to Opus 5. The QFBench curator for Sonnet 4.6 stops before submitting, so its package is its last complete skill under the rule in table 20. Each cell is one trial with Claude Code as the harness. Reward is the task verifier's pass or fail on eight tasks, and the fraction of checks passed on `crystallographic-wyckoff-position-analysis` and `lab-unit-harmonization`. An agent timeout scores 0; build and sandbox failures are rerun and excluded. A skill counts as opened when the Claude Code transcript records a call to that skill's tool.

Table 21: *Every pilot trial. None, human (the task's own SkillsBench skills) and ours (our curator's package for the matched domain).* ^{o}*: the learner opened the skill;* ^{t}*: agent timeout, scored* 0*.*

		Sonnet 4.6			Opus 5		
Task	Domain	None	Human	Ours	None	Human	Ours
`econ-detrending-correlation`	Finance	1	1°	1°	1	1°	1°
`invoice-fraud-detection`	Finance	1	0°	0	0	1	0
`sec-financial-report`	Finance	0	1°	0	0	1°	0°
`shock-analysis-demand`	Finance	0^{t}	$0^{t,o}$	0^{t}	0	0°	0
`weighted-gdp-calc`	Finance	0	1°	0	1	1°	1
`crystallographic-wyckoff-position-analysis`	Science	1	1°	.45	.45	.36	.45
`earthquake-plate-calculation`	Science	0	1°	0	0	1°	0
`protein-expression-analysis`	Science	0	1°	0	0	1°	0
`lake-warming-attribution`	Science	0	1°	0	0	1°	0
`lab-unit-harmonization`	Clinical	.69	$.50^{o}$	.65	0^{t}	1°	0^{t}

G OUR BASELINES AGAINST PUBLISHED SCORES

Our skillless baselines differ from leaderboard scores because the protocols differ. Table 22 compares them with published numbers verified at the primary source and records the protocol differences. *None* means that no verified published number exists for that model and benchmark. Every lift in this paper uses our own paired baseline. The largest differences arise from subsets chosen for headroom, the absence of web access on HLEBench, and the retrieval tools and user simulator on τ^3 banking.

Table 22: *Published scores against our skillless baselines. Published protocols: HLE from the Claude Sonnet 4.6 system card (full 2,500 questions, graded by Claude Sonnet 4.5; with tools means web search, web fetch and code execution); τ^3 banking from the Sierra leaderboard (pass^1, all retrieval tools, GPT-5.2 user simulator, maximum effort, four trials). Ours: the chemistry, engineering and CS/AI categories only, bash and code but no web, a turn cap, timeouts scored as failures, judged by* `claude-sonnet-4-6`*; τ^3 banking with keyword retrieval only and a* `claude-sonnet-4-6` *user simulator on* 30 *held-out tasks.*

Benchmark	Model	Published	Ours (range over runs)	Main differences
HLE	Sonnet 4.6	0.33 no tools, 0.49 with tools	0.17 to 0.18	subset, no web, timeouts, judge
HLE	Opus 5	none	0.44 to 0.50	
HLE	GLM 5.3 Flash	none	0.09 to 0.18	
τ^3 banking	Opus 5	0.49	0.20 to 0.43	retrieval, simulator, effort, $n = 30$
τ^3 banking	Sonnet 4.6	none	0.13 to 0.23	
τ^3 banking	GLM 5.3 Flash	none	0.00 to 0.03	plain-text turns end episodes
HealthBench Hard	all three	none	see table 20	three of seven themes, Haiku judge

H TRANSFER: EVERY TARGET AGAINST EACH SKILL PICK

The atlases compare shipped skills with the source learner's best saved checkpoint on every evaluation learner. Figure 4 shows only shipped skills. For each atlas, the second pick is the checkpoint with the highest source learner held-out score under the stated selection metric. Because the curator never sees those scores, this pick measures the transfer of an available checkpoint rather than a choice the curator can make. The atlases cover earlier HealthBench and QFBench source runs whose checkpoints are evaluated on every target. The newer HLEBench and QFBench transfers in fig. 4 evaluate only shipped skills and appear in `figures/results-20260925.json`. These atlases summarise the curves; `figures/transfer_numbers.json` gives every plotted value, paired sample size and bootstrap interval.

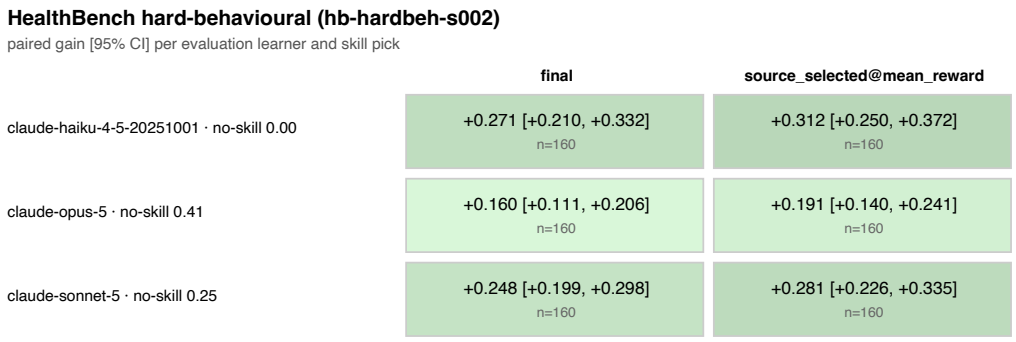

Figure 5: *HealthBench `hb-hardbeh-s002`: every evaluation learner with the shipped skill and the source learner's highest-scoring checkpoint.*

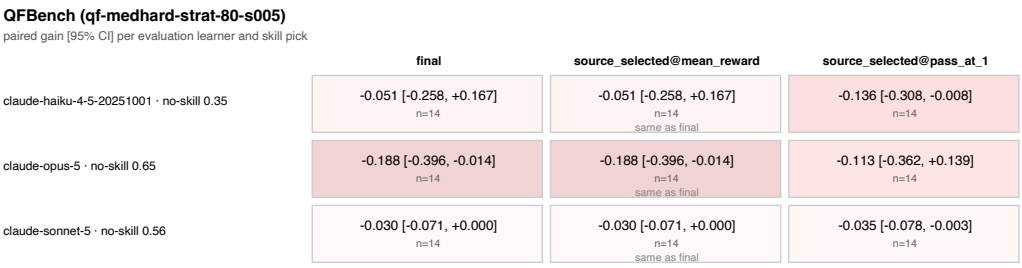

Figure 6: *QFBench `qf-medhard-strat-80-s005`: every evaluation learner with the shipped skill and the source learner's highest-scoring checkpoints.*

I A STUCK LOOP BURNS THE BUDGET

The OpenHands×GLM 5.3 Flash learner exhausts its HLEBench curation pool because its turn protocol repeatedly resends long prompts (table 9). On the same split and curator, it uses 299.7M of 300M tokens across two probes and 84 task runs, while Claude Code×Sonnet 4.6 uses 238.5M of 500M across eight probes and 343 of 412 training tasks. This is 3.57M against 0.66M tokens per probe task, a $5.4\times$ difference. OpenHands expects a tool call on every turn. When GLM reasoning exceeds its output cap, it returns neither tool call nor content: 174 of 3,464 calls end with `finish_reason: length`, and median completion is 150 tokens against Sonnet's 1,094. The harness then prompts it again, resending the conversation. Initial prompts are twice as large (median 65k against 33k), and calls per task are $2.7\times$ as frequent (41 against 15). Their product matches the $5.4\times$ token difference. The first 60-task probe alone costs 197M, two thirds of the pool, because batch size is calibrated to Sonnet's task cost. Caching reduces dollar cost but not metered tokens: 96.6% of spending is cache reads, which the gateway counts in full. The same failure appears in

held-out evaluation (table 10): 38 of 105 baseline trials return no agent output and score zero. The shipped skill removes these failures by requiring the first tool call to write a placeholder response file and prohibiting consecutive reasoning-only turns. No-output trials fall from 38 to 0, and mean reward rises from 0.0857 to 0.1429 ($1.67\times$). A per-task turn cap (the measured mean is 41 calls) and skill instructions against repeated reasoning turns address this failure. With only two small probes, the curator's best-iteration declaration rests on trace evidence rather than measured deltas and states that limitation.

J Curator Strategy and the Benchmark It Meets

Different probing strategies produce similar QFBench results but different HLEBench skills. Fable 5 favours coverage: on QFBench split `s005`, nine probes cover all 54 development tasks in 108 trials, with retries concentrated late; on HLEBench it covers 162 unique tasks without replication. Opus 5 favours paired comparisons across skill versions: it repeats a fixed 60-task HLEBench panel three times (93 unique tasks in 343 trials) and covers 45 of 54 QFBench tasks. On QFBench, both 100M final skills reach similar held-out reward, 0.687 and 0.690 (table 16). Their apparent lift gap, $+20.1$ against $+5.9$ points, reflects baseline variation: four skillless runs of the same configuration span 0.472 to 0.627 (sd 0.073), so lifts across runs with $n = 14$ are not comparable. On HLEBench, Fable's two-file skill combines process rules and a chemistry reference sheet, but its detailed procedure increases timeouts. Its final reward is 1.9 points below baseline (table 11). Opus provides five reference files across computer science and AI, engineering, and chemistry. The learner reads them rather than deriving the material again, and held-out reward rises by 14.3 points with faster execution (table 8). Broad coverage helps on a compact domain with aligned verification. On the heterogeneous judged set, scores range from 0.125 to 0.5 without a repeated panel to reduce noise, while repeated comparisons provide more useful evidence.