LONG-HORIZON-TERMINAL-BENCH 2.0: TESTING THE LIMITS OF AGENTS ON LONG-HORIZON TERMINAL TASKS WITH DENSE REWARD-BASED GRADING

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents have become increasingly capable of autonomously completing short and well-specified terminal tasks. However, existing terminal benchmarks either focus on short simple problems that finish within a few minutes and or hard tasks, which are typically evaluated only by their final outcome. This setup overlooks intermediate progress and partial solutions, leading to sparse reward signals and an incomplete picture of distinguishing agent capability. We curate LONG-HORIZON-TERMINAL-BENCH 2.0 (LHTB 2.0), a diverse set of 77 long-horizon sweep grouped into 13 domains. Each task follows a Terminal-Bench-style harbor setup with a reference solution or simulation engine, and is further decomposed into fine-grained graded subtasks. We design dense intermediate rewards and partial assignment credit, allowing evaluation to capture not only whether an agent reaches the final goal, but also how far it progresses on long-horizon workflows.

We evaluate 15 frontier models on LHTB 2.0, the strongest current system is GPT-6-astra, which reaches soft pass threshold ($R \geq 0.95$) on $37/77$ tasks (48.1%) and achieves dense mean reward 0.68, followed by GPT-5.6-sol at $32/77$ tasks (41.6%) and mean reward 0.67. Across 1,145 scored task runs, only 16.8% pass the soft pass $R \geq 0.95$ threshold, while 15.4% make no meaningful progress ($R < 0.05$) and 67.9% achieve partial-progress. These results reveal substantial headroom for improvement. We further analyze common failure modes and error patterns to evaluate models and agents on long-horizon terminal tasks.

1 INTRODUCTION

AI agents built on large language models (LLMs) are rapidly improving at autonomous decision making and tool use (Du et al., 2026; Chowa et al., 2026). Recent work has demonstrated impressive performance on short, well-scoped tasks such as fixing a code repository issue, completing a coding ticket, or issuing a handful of shell commands (Merrill et al., 2026). However, they still cover only a narrow slice of the long-horizon, domain-specific, and practically important workflows that human experts care about in practice (Zhou et al., 2023; Xie et al., 2024; Merrill et al., 2026).

Nowadays, many real workflows are *long-horizon*: they require agents to execute hundreds of steps, maintain and update plans over tens of minutes to hours, and manage evolving long-context state (Wang et al., 2026). Examples include reproducing results from published research papers (Song et al., 2026), installing environments from a github repo, auditing complex multimodal datasets, debugging compiler toolchains, or shepherding a multi-stage ML training pipeline. In these settings, agents must repeatedly use terminal command lines to read and write files, run scripts, inspect partial outputs, revise their plans, and recover from mistakes. The outcome is determined not by a single action, but by sustained, coherent progress over a long sequence of decisions (Liu et al., 2026).

Existing benchmarks lack fine-grained verifiers to evaluate agents on long-horizon terminal tasks (Jimenez et al., 2024). Dominant terminal and software-engineering benchmarks typically evaluate within a few minutes to at most an hour, and grade agents solely on whether the final state satisfies a test suite (Desai et al., 2026; Chu et al., 2026). This leaves two gaps. First, short tasks or narrow horizons understate the difficulty of real workflows that require long-horizon navigation,

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.

errored step revision, and multi-stage debugging, not just local patching (Merrill et al., 2026; Feng et al., 2026). Second, outcome-only grading yields extremely sparse reward signals: an agent that completes most intermediate steps but fails at the final step can receive the same score as one that fails from the beginning (Lin et al., 2025). Both issues make it difficult to understand how well current agents can sustain progress on long workflows, or where they fail.

In this paper, we introduce LONG-HORIZON-TERMINAL-BENCH 2.0 (LHTB 2.0), a 77-task long-horizon benchmark spanning across 13 domains such as e.g, natural science, engineering, scientific computing. Every task is hosted in a containerized terminal environment and decomposed into graded subtasks with intermediate checks. Rather than asking agents to `solve everything or fail`, the grader assigns credit for partial progress toward a final goal, providing dense feedback along long trajectories while preserving the realism of everyday tasks.

Our validation experiment contains 1,145 scored model–task cells across 15 frontier models. On the dense all-77 basis, GPT-6-astra (OpenAI, 2026) is the strongest model with 37 tasks at $R \geq 0.95$ and mean reward 0.68, followed by GPT-5.6-sol (OpenAI, 2025b) with 32 tasks at $R \geq 0.95$ and mean reward 0.67. Kimi K3 (Kimi, 2026) is a distant third by dense mean reward at 0.55. Even after dense regrading, the full benchmark remains far from saturated: only 192 of 1,145 scored task runs (16.8%) pass at reward threshold $R \geq 0.95$, while 176 (15.4%) make no meaningful progress and 777 (67.9%) remain in the partial-progress regime. In other words, more than two thirds of the observed runs would still be counted as total failures under binary grading despite accumulating nontrivial reward, suggesting that long-horizon execution remains a central bottleneck for current models.

We also conduct qualitative and quantitative analyses of agent failures on LHTB 2.0. Our analysis shows that current agents often fail not because every local step is wrong, but because they cannot reliably sustain progress, verify completion, and finish long-horizon tasks within budget. Dense rewards make these differences visible by separating timeout-driven incomplete progress from premature stopping and weak self-verification. Detailed termination-cause and domain-level analyses are provided in Appendix B. We will release LHTB 2.0 and the accompanying evaluation harness to support future work on agents that can plan, verify, and execute reliably over long horizons.

2 LONG-HORIZON-TERMINAL-BENCH 2.0

LONG-HORIZON-TERMINAL-BENCH 2.0 builds on the Terminal-Bench formulation (Merrill et al., 2026), where each task is a containerized terminal environment with a natural-language instruction, all relevant files and tools, and a reference solution that can be executed entirely from the command line. Tasks span diverse domains and are explicitly designed so that progress requires *continued* work over many steps: reading and modifying code and data, running long scripts, inspecting partial outputs, and iteratively debugging.

Unlike Terminal-Bench, which evaluates tasks in a binary solved/unsolved fashion, LONG-HORIZON-TERMINAL-BENCH 2.0 is graded through a subtask-based scheme. Each task is decomposed into a small set of semantically meaningful subtasks with their own checks, and agents are scored by their completion rate across these subtasks. This design provides partial credit and a much denser signal about long-horizon performance, revealing how far agents get through complex workflows rather than only whether they reach the final goal.

2.1 TASK FORMULATION

A LONG-HORIZON-TERMINAL-BENCH 2.0 task follows the Terminal-Bench formulation (Merrill et al., 2026): it is specified as a Harbor task with (1) a natural-language instruction, (2) a Docker image that defines the terminal environment, (3) a task configuration file, and (4) an oracle implementation or simulator used for grading. The instruction describes the overall long-horizon goal (e.g., reproduce a figure, repair a robotics SLAM pipeline, audit a multimodal dataset) and is the only specification the agent sees. The Docker image contains all relevant assets, code, data, tools, and helper scripts, so that an agent can complete the task entirely from the terminal. Tasks are interactive: after the container starts, the agent issues shell commands, edits files, runs scripts, and inspects intermediate outputs over hundreds of steps until it either succeeds or times out.

While the task format matches Terminal-Bench 2, LONG-HORIZON-TERMINAL-BENCH 2.0 is explicitly designed so that a single subtask typically requires many reasoning steps and many minutes to hours of work and dozens to hundreds of distinct operations. Long-horizon structure comes both from the underlying domains (multi-stage ML pipelines, campaign-style games, scientific audits) and from the way we factor goals into a sequence of intermediate targets that must be discovered and executed over time.

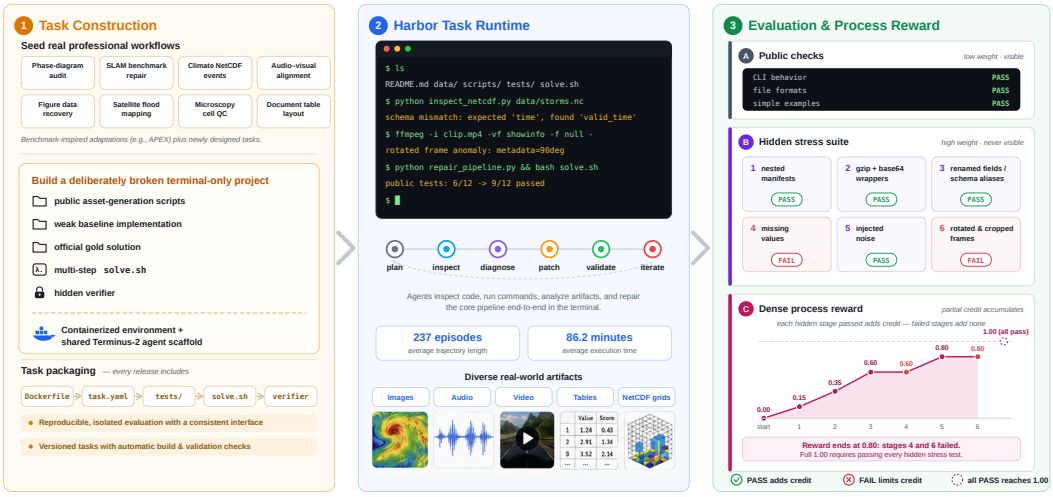


Figure 1: The overall structure of long-horizon terminal task with dense reward grading. Partial rewards show how far models are into completing the task.

2.2 SUBTASK-BASED GRADING

Each LONG-HORIZON-TERMINAL-BENCH 2.0 task is graded by a deterministic grader that runs *inside* the container at the end of a rollout. Instead of returning a strict single pass/fail, the grader computes scores for a small set of subtasks $\{s_1, \ldots, s_K\}$ that together describe the intended workflow. For every subtask s_k it outputs a normalized score $r_k \in [0, 1]$; the task reward is

$$R = \frac{\sum_{k=1}^{K} w_k \, r_k}{\sum_{k=1}^{K} w_k},$$

where w_k are non-negative weights (by default all equal, with higher weight on the final goal when needed). Subtasks are chosen to correspond to meaningful intermediate goals in the workflow, and each is evaluated using objective evidence from the final container state, such as files, outputs, test results, or simulator responses.:

- **Binary subtasks.** Checks are implemented as strict Boolean conditions over the environment state, such as whether all unit tests exit successfully, whether a service responds on the expected port, or whether required experiment scripts run without error. These subtasks return $r_k \in \{0, 1\}$ depending on whether the corresponding programmatic check passes.

- **Continuous or thresholded subtasks.** For quantitative targets (e.g., reproducing a figure, matching a metric, or achieving a given speedup) we use continuous scores. Examples include *1.0 if the reproduced metric is within tolerance of the reference, decreasing linearly to 0 as error grows*, or *fraction of held-out examples on which the model's predictions match the oracle*. This yields meaningful partial credit when an agent gets close but not perfect.

- **Episode-aggregating subtasks.** Campaign-style tasks (e.g., games or repeated audits) aggregate across episodes. A typical subtask is the fraction of episodes or levels in which the environment's internal success flag is triggered, or the mean normalized reward reported by the simulator. This measures whether an agent can perform a long-horizon behavior reliably across many episodes, rather than succeeding only once by chance.

The overall task reward R is used for all reported metrics. A task is counted as *resolved* for sucess if $R \geq \tau$[1]. We also report mean R across tasks to capture how far agents progress on problems they cannot fully solve.

2.3 Dataset construction

We construct LHTB 2.0 by instantiating a common recipe across a diverse set of real-world long-horizon workflows (Vidgen et al., 2026; Wu et al., 2026). Each task begins with a realistic professional data-processing or engineering problem, such as materials phase-diagram auditing, robotics SLAM benchmark repair, climate NetCDF extreme-event detection, audio–visual event alignment, scientific-figure data reconstruction, satellite flood change detection, microscopy cell-count quality control, or scanned-document table reconstruction. Some of these professional-workflow tasks are adapted from or inspired by prior benchmarks such as APEX-Agents (Vidgen et al., 2026).



Figure 2: **Task distribution across the 13 LHTB 2.0 domains.** The benchmark covers e.g, Earth, climate & energy and Systems, performance & security, Multimodal & imaging analysis and Software & reverse engineering tasks.

For each problem, we build a complete but deliberately broken terminal-only project. Agents must work entirely through the terminal: inspecting code, running commands, and analyzing artifacts such as images, audio, video, tables, and NetCDF files in order to diagnose and repair the core pipeline. Each task includes reproducible public asset-generation scripts, a weak baseline implementation, an official gold solution, a multi-step `solve.sh`, and a hidden verifier.

To discourage overfitting to superficial public tests, the public checks only validate command-line behavior, file formats, and a small number of simple examples, and they carry relatively low reward weight. Most of the reward is assigned through hidden stress cases that dynamically generate harder inputs and schema variations, such as nested manifests, gzip-plus-base64 wrappers, renamed fields, missing values, injected noise, rotated or cropped images, anomalous frames, and alternative coordinate or time-dimension conventions. As a result, agents cannot obtain high scores by hard-coding outputs or by only patching the public cases; they must implement robust parsing, core algorithms, and end-to-end artifact generation that generalize beyond the visible examples. The official gold solution is required to achieve a verifier score of 1.0 on the full hidden evaluation suite.

Difficulty calibration and validation. We calibrate difficulty by repeatedly running Deepseek-V4-Pro (DeepSeek-AI, 2026) under 3-hour time budgets and adjusting task design until the tasks are challenging but still solvable in principle. We also use DeepSeek-V4-Pro to validate alignment between each task's instructions and its verifier, checking whether the instructions omit any essential details that a model would need in order to pass without relying on hidden information. We curated a total of 77 tasks, which are implemented in the Harbor format (Merrill et al., 2026), with containerized environments and a shared agent harness.

[1] we use a soft threshold such as $\tau = 0.95$ or strict threshold $\tau = 0.95$.

2.4 TASK COMPOSITION

The 77 tasks cover 13 domains: Geospatial & cartography; Document, OCR & data engineering; Life sciences, chemistry & materials; and Physical simulation, signals & control. Figure 2 shows the resulting distribution. The largest groups are Earth, climate & energy and Systems, performance & security with 9 tasks each, followed by Multimodal & imaging analysis and Software & reverse engineering with 8 each. The remaining categories contain between 3 and 7 tasks.

3 EXPERIMENTS

Harness and agents. We evaluate models using the Harbor framework with the Terminus-2 agent harness (Harbor Framework Team, 2026), which interacts with each task environment through a single long-horizon terminal session. The consistent harness framework allows us to evaluate how existing frontier models perform on sustained terminal-based workflows fairly.

Models. We benchmark GPT-6-astra, GPT-5.6-sol (OpenAI, 2025b), DeepSeek V4 Pro (DeepSeek-AI, 2026), DeepSeek V4 Flash, Gemini 3.1 Pro (Google DeepMind, 2026), Gemini 3.8 Flash, GLM-5.3, GLM-5.2, Kimi K3, Kimi K2.6 (Moonshot AI, 2026), MiniMax M3 (MiniMax, 2026), Qwen 3.8 Max, Doubao Seed 2.1 Pro (ByteDance Seed Team, 2026), Hunyuan hy4, and Grok-4.6.

Metrics. For each model, we report the soft pass rate at $R \geq 0.95$, the strict pass rate (the fraction of tasks with normalized reward exactly 1.0), the mean normalized reward across all tasks, the number of episodes required per task, the elapsed runtime per task, the estimated dollar cost per task, and the token usage per task.

3.1 MAIN RESULTS

We set the timeout limit for each task to be 3 hours for cost reasons. Figure 3 summarizes pass rate at $R \geq 0.9$, $R \geq 0.95$, and $R \geq 1.0$, along with mean reward across all 77 tasks. Overall, the evaluated frontier models still struggle on long-horizon tasks, even under a dense-grading evaluation that gives credit for partial progress. GPT-6-astra is the strongest reported model, but still resolves only 48.1% of tasks (37/77) at $R \geq 0.95$. GPT-5.6-sol follows at 41.6% (32/77). Kimi K3 is a distant third at 22.1% (17/77). A second tier includes hy4 and Gemini 3.8 Flash at 19.5% (15/77) each, followed by GLM-5.3 at 15.6% (12/77). Below that, DeepSeek V4 Flash and Grok-4.6 resolve 13.0% (10/77) each; DeepSeek V4 Pro reaches 11.7% (9/77); GLM-5.2 and Doubao Seed 2.1 Pro reach 10.4% (8/77) each; and Qwen 3.8 Max reaches 9.1% (7/77). MiniMax M3 and Gemini 3.1 Pro resolve 6.5% (5/77) each, while Kimi K2.6 resolves only 2.6% (2/77). Mean normalized rewards follow a similar but not identical ordering: GPT-6-astra also leads this metric at $R = 0.68$, while Gemini 3.1 Pro records the lowest mean reward ($R = 0.40$) among the reported models. The gap between pass rate and mean reward for several mid-tier systems indicates that many models make substantial partial progress on hard tasks without fully resolving them. These results suggest that long-horizon execution is not merely a local reasoning problem, but remains a central bottleneck even for top frontier models.

In addition, in the open-source models, model performance broadly tracks release date: earlier releases such as Kimi K2.6 (April 20, 2026) and GLM-5.2 (June 16, 2026) are surpassed by later releases such as Kimi K3 (July 16, 2026; open weights July 27), GLM-5.3 (August 14, 2026; open weights August 28), and Hunyuan hy4 preview (August 28, 2026).

3.2 DENSE REWARD IS NECESSARY TO RANK MODELS

A key contribution of LONG-HORIZON-TERMINAL-BENCH 2.0 is dense, subtask-level grading rather than binary pass/fail, which is essential at this difficulty level. Under a strict threshold ($R \geq 1.0$), pass rates fall substantially across the board (Figure 3); even at $R \geq 0.95$, only 16.8% of runs pass while 15.4% score below 0.05 (Figure 4). The remaining 67.9% of runs make real partial progress, but binary grading would count them as complete failures by giving them the same score as a run that makes no progress.

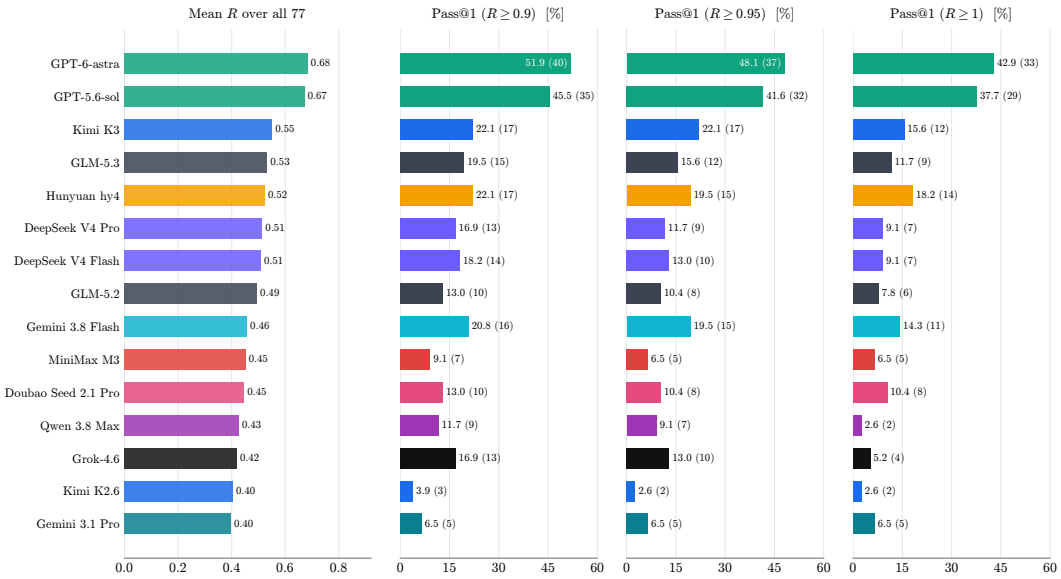


Figure 3: **LHTB 2.0 leaderboard.** Pass rate at reward thresholds 0.9, 0.95, and 1.0, together with mean normalized reward, for each model under a shared agent setup, shown in a single model order sorted by mean reward across all 77 tasks. Pass rate is the fraction of the 77 tasks whose overall reward R meets a threshold τ, with $R \geq 1.0$ corresponding to full completion, while mean R captures partial progress across all tasks. GPT-6-astra achieves the strongest overall performance, resolving 48.1% of tasks at $R \geq 0.95$. Tightening the threshold to $R \geq 1.0$ lowers pass rates across nearly the entire board, showing that partial-credit evaluation captures progress that a binary threshold misses.

Binary grading hides meaningful differences between models. Under binary pass/fail evaluation, the leaderboard still collapses into tie groups; for example, Hunyuan hy4 and Gemini 3.8 Flash each solve 15 tasks at the $R \geq 0.95$ threshold. Mean reward resolves these ties by quantifying how far a model progresses on tasks it does not fully complete. Among the models tied at 15 solved tasks at $R \geq 0.95$, mean reward spans from 0.46 for Gemini 3.8 Flash to 0.52 for Hunyuan hy4. Pass rate and mean reward are positively but only moderately correlated (Spearman $\rho = 0.85$), and they can produce different model rankings because they measure different aspects of long-horizon performance: pass rate captures full task completion, whereas mean reward captures sustained partial progress.

Near-misses reveal real progress that binary pass/fail evaluation would hide. Dense grading also reveals *near-misses* ($0.75 \leq R < 0.95$), runs that are close to completion but fail some final checks. These near-misses still occur frequently (121 runs), indicating that many trajectories get close to solving the task but fail at the last few steps or checks. For example, Kimi K2.6 solves only two tasks at the $R \geq 0.95$ threshold, yet has seven near-misses and a mean reward of 0.40. Its strongest near-miss reaches $R = 0.92$ on `matpower-opf-regression`, and it also comes close on `poc-exploit-craft` ($R = 0.89$), `apex-openroad-ibex-signoff` ($R = 0.89$), and `generals-bot-arena` ($R = 0.85$). Under binary grading, it would appear substantially weaker than its dense-reward profile suggests, even though it repeatedly approaches successful completion. As models improve, such near-misses are likely to provide an important early signal of progress, often revealing clear capability gains before they are fully reflected in pass rates.

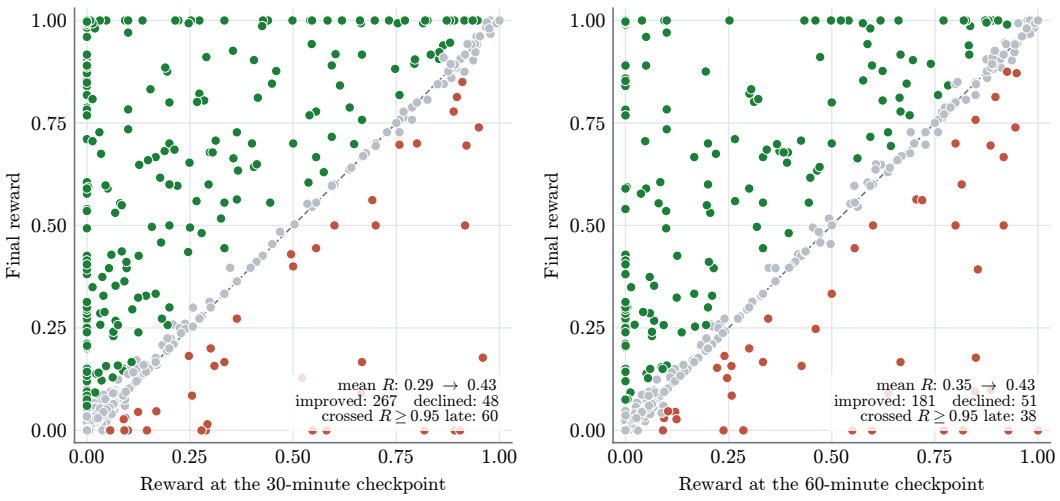


Figure 5: **Within-run reward change after 30 and 60 minutes.** Each point represents one scored-task run: the x-axis gives its reward near the indicated checkpoint, and the y-axis gives its final reward. The dashed diagonal denotes no change. Green points finish more than 0.05 higher, red points finish more than 0.05 lower, and gray points remain within that margin. Aggregate statistics are shown in the lower-right corner of each panel.

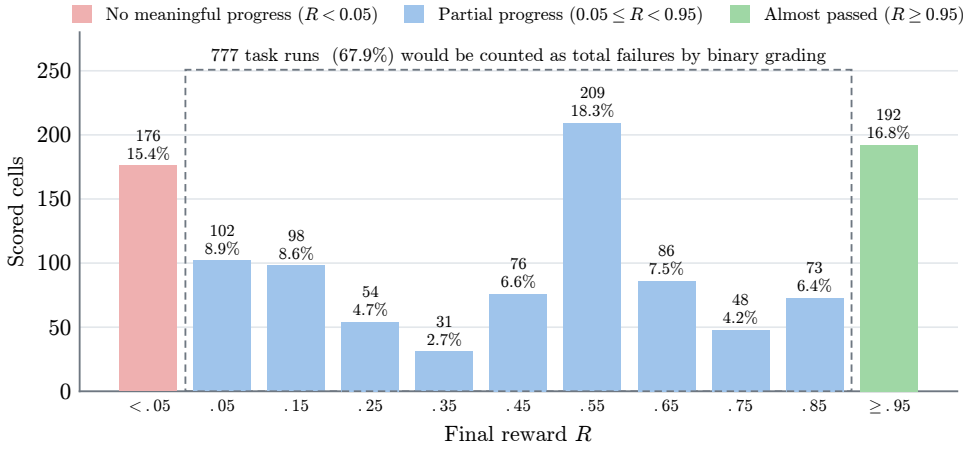


Figure 4: **Distribution of final interval rewards over all scored LHTB 2.0 runs.** Across 1,145 scored task runs, 192 (16.8%, green) pass the $R \geq 0.95$ threshold, 176 (15.4%, pink) make no meaningful progress ($R < 0.05$), and 777 (67.9%, blue) land in the partial-progress regime. The dashed box emphasizes that these 777 task runs would all be counted as total failures under binary pass/fail evaluation despite covering a wide range of substantial partial progress; 485 runs (42.4%) still reach $R \geq 0.5$, and the mass in $[0.85, 0.95)$ contains 73 near-misses.

Detailed token usage, runtime, and cost-efficiency results are reported in Appendix A.

3.3 Time and Process Reward Analysis

A key benefit of process rewards is that they reveal how model–agent systems progress throughout a long-horizon task, rather than evaluating only the final outcome.

Figure 5 compares final reward with rewards observed after 30 and 60 minutes. At 30 minutes, mean reward increases from 0.29 to 0.43 by the end of the run; 267 of 693 runs improve by more

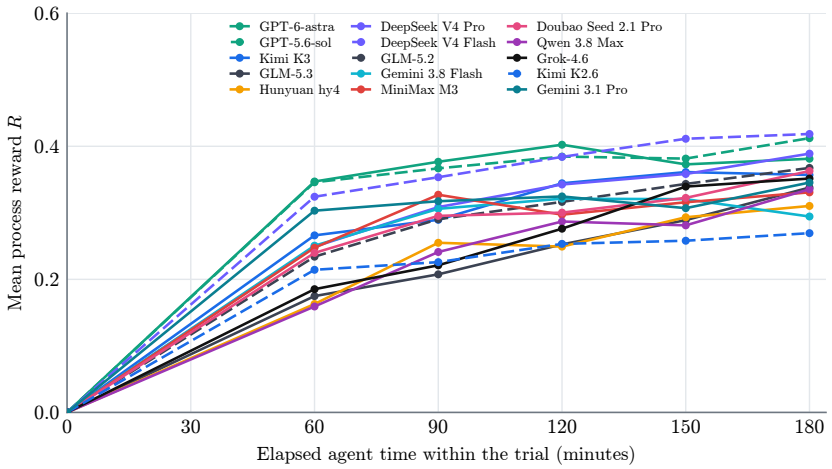


Figure 6: **Process reward over time on unresolved released-task cells.** Each curve includes only released-task runs whose final reward remains below 0.95. Curves start at $(0, 0)$ by definition and omit the sparsely populated 30-minute interval. Within each time bin, checkpoints are first averaged per task and then across tasks, so every task receives equal weight regardless of how many checkpoints it produced.

than 0.05, and 60 cross the soft-pass threshold after that checkpoint. At 60 minutes, mean reward increases from 0.35 to 0.43; 181 runs still improve by more than 0.05, and 38 subsequently cross the soft-pass threshold. The later checkpoint is naturally closer to the final score, but the right panel shows that substantial progress and many successful completions still occur after the first hour.

We analyzed the 65 unique model–task trajectories with final rewards more than 0.05 below a 30- or 60-minute checkpoint (48 and 51 red points, respectively, with 34 overlapping). In 55 cases, later checkpoints also decline, confirming within-run regression; 54/65 end at the time limit. Most regressions occur in stateful games and puzzles (24) or artifact-producing and reproduction tasks (21), where continued actions can worsen the state or overwrite working outputs. The remaining 10 drops appear only during final verification and may reflect last-minute edits or verifier variation. Thus, red points indicate lower final scores, but not uniformly model regression.

Figure 6 asks a stricter question: whether runs that never pass still make progress. After successful runs are removed, every model's mean reward at 180 minutes remains above its 60-minute level, despite intermediate fluctuations, and the strongest models generally sustain higher process rewards throughout. Unresolved runs are therefore not uniformly static failures; many continue improving for hours without closing the final gap.

Detailed analyses of termination causes and category-level difficulty are provided in Appendix B.

4 Related Work

Terminal and software-engineering agent benchmarks. Recent agent benchmarks have shifted from static question answering and isolated programming exercises toward interactive, execution-grounded tasks in realistic work environments (Zhou et al., 2023; Xie et al., 2024). In software engineering, SWE-Bench (Jimenez et al., 2024) evaluates repository-level issue resolution on real GitHub projects, while LiveCodeBench (Jain et al., 2024) provides contamination-aware evaluation of code generation and related coding abilities such as execution, self-repair, and test-output prediction. Subsequent benchmarks further broaden the scope of software-agent evaluation: SWT-Bench (Mündler et al., 2024) studies test generation and validation for real-world bug fixes, and Terminal-Bench (Merrill et al., 2026) places agents in containerized terminals to complete realistic technical tasks such as system configuration, model training, and research-code reproduction. More recent efforts push toward substantially harder and longer workflows, including Frontier-SWE (Chu et al., 2026), SWE-EVO (Le et al., 2025), SWE-Marathon (Desai et al., 2026), and Edge-Bench, which target expert-level, multi-stage, or ultra-long-horizon tasks. These benchmarks

measure progress in coding and terminal agents, but many still primarily measure end-to-end success or failure, providing limited visibility into how much useful progress an agent made before failing. LONG-HORIZON-TERMINAL-BENCH 2.0 uses subtask-based grading exposes partial progress even when the final objective is not fully completed.

Measuring long-horizon autonomy tasks. Recent work has increasingly characterized long-horizon autonomy in terms of the duration of tasks that agents can complete reliably, rather than aggregate performance on a fixed benchmark (Kwa et al., 2025; Wijk et al., 2024). METR's time-horizon analysis (Kwa et al., 2025) makes this perspective explicit by estimating the human-time duration of tasks that frontier agents can complete at a fixed success rate. This framing treats horizon length as a first-class capability variable and is grounded in human-calibrated evaluations such as RE-Bench (Wijk et al., 2024) and HCAST (Rein et al., 2025), which compare agents against human performance on realistic research-engineering, software-engineering, cybersecurity, and reasoning tasks. At the same time, it also highlights a limitation of outcome-only evaluation in long-horizon settings: a binary success signal provides little information about how much progress was made before failure or where execution began to diverge. Recent studies show that long-horizon degradation can arise from the accumulation of execution errors over extended trajectories, even when short-horizon competence appears strong (Sinha et al., 2025). LONG-HORIZON-TERMINAL-BENCH 2.0 is designed to complement this line of work by combining extended, many-step autonomy tasks with subtask-level grading, thereby enabling a finer-grained characterization of partial progress and failure throughout long-horizon execution.

Beyond outcome-only grading. Process reward models provide supervision over intermediate reasoning steps rather than only final answers, yielding denser signals than outcome-only supervision (Lightman et al., 2023; Khalifa et al., 2025). In parallel, rubric-based approaches decompose evaluation into individually assessable criteria or weighted dimensions of quality, making it possible to score complex outputs beyond a single holistic judgment (Rao & Callison-Burch, 2026; Pan et al., 2026; Tian et al., 2026). Most of this literature uses fine-grained signals primarily as *training* rewards or judge-based supervision, often relying on learned verifiers or LLM evaluators (Uesato et al., 2022; Zheng et al., 2023; Li et al., 2025; Yuan et al., 2024). LONG-HORIZON-TERMINAL-BENCH 2.0 applies the same core intuition at *evaluation* time: each task is decomposed into semantically meaningful subtasks, and each subtask is checked by deterministic, environment-grounded graders.

Agent harnesses. Benchmark outcomes depend not only on the underlying model, but also on the agent harness that mediates its interaction with the environment (Yao et al., 2026; Yang et al., 2024; Li et al., 2026; Lin et al., 2026). Open frameworks such as SWE-agent (Yang et al., 2024), OpenHands (Wang et al., 2024), and the Terminus family introduced alongside Terminal-Bench (Merrill et al., 2026) provide reusable interfaces for autonomous repository- and terminal-based work. Open-source agents such as Codex (OpenAI, 2025a) and OpenClaw (OpenClaw Contributors, 2026) package nontrivial scaffolding around the base model, including prompting, tool orchestration, context management, and action-selection policies.

5 CONCLUSION

We introduce LONG-HORIZON-TERMINAL-BENCH 2.0, a benchmark of 77 containerized terminal tasks spanning 13 domains and designed to evaluate long-horizon execution. Its deterministic, environment-grounded subtasks provide dense rewards that measure both task completion and partial progress. Across 15 frontier models evaluated with a shared `terminus-2` agent, the strongest model, GPT-6-astra, resolves only 48.1% of tasks ($37/77$) at $R \geq 0.95$, while only 192 of 1,145 scored runs (16.8%) pass across the fleet. Nevertheless, 777 runs make meaningful but incomplete progress, underscoring the information lost under binary grading. Each task requires an average of 262.5 LLM calls, 27.3M tokens, 94.3 minutes, and an estimated $48.4 (Appendix A). The primary bottleneck is sustained execution rather than local reasoning alone. Most unresolved runs exhaust the time budget after making partial progress; many successful runs complete only late in the trajectory, while others lose earlier gains by overwriting working artifacts. Detailed failure-mode and domain-level analyses are provided in Appendix B. Future agents therefore need stronger long-horizon planning, memory and progress tracking, self-verification, checkpoint recovery, and stopping decisions that preserve the best available solution.

AI USE STATEMENT

Generative AI tools were used both as the systems evaluated in this work and as research assistants. We used these tools to assist with checking alignment between task instructions and verifiers, implementing and debugging evaluation and analysis code, processing data, analyzing and interpreting results, creating and revising figures, and drafting and editing the manuscript. In particular, DeepSeek V4 Pro was used to help inspect task–verifier alignment. All AI-assisted code, analyses, figures, and text were reviewed by the authors; reported statistics were checked against the frozen result exports and the corresponding analysis scripts were rerun. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work evaluates AI agents in isolated, containerized terminal environments and does not involve human subjects or private personal data. Some tasks cover security-relevant workflows; these tasks operate only on synthetic or local resources and are intended for controlled capability evaluation, not for use against real systems. The task environments contain no real credentials or production targets. We report aggregate model performance and document the computational resources and estimated costs associated with the evaluation. The benchmark and evaluation harness will be released with documentation describing their intended research use and applicable licenses.

REPRODUCIBILITY STATEMENT

Section 2 describes the task format, dense grading scheme, benchmark construction, and category composition. Section 3 specifies the evaluated models, agent harness, time budget, and metrics, while Appendix A describes the cost-estimation procedure. Section 3.3 presents the checkpoint analysis, Appendix B provides additional failure-mode and domain-level results, and Appendix C gives the complete inventory of 77 tasks. The anonymized supplementary material contains the evaluation configurations, frozen result exports, dense-regrading data, cost-estimation code, and scripts used to generate the tables and figures. Upon acceptance, we will release the full benchmark tasks, containerized environments, verifiers, evaluation harness, and agent trajectory outputs.

REFERENCES

ByteDance Seed Team. Seed2.1 officially released: Advancing ai productivity. https://seed.bytedance.com/en/blog/seed2-1-officially-released-advancing-ai-productivity, June 2026. Official model release announcement (Doubao Seed 2.1 / Seed 2.1 Pro).

Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 2026.

Evan Chu, Rajan Agarwal, Abishek Thangamuthu, Brendan Graham, Justus Mattern, Freeman Jiang, Paul Cento, Swarnim Jain, Mersad Abbasi, Mohammad Hossein Rezaei, George Wang, Alex Zhang, Simon Guo, Karina Nguyen, Danna Liu, Arash Bidgoli, Aditya Dalmia, Apoorv Dankar, Ashrut Vaddela, Calvin Chen, Keshav Kumar, Kushagra Vaish, Navid Pour, Rishyanth Kondra, Sagar Badiyani, Sidharth Giri, Snagnik Das, Soham Gaikwad, Syed Shah, Vagish Dilawari, and Vishal Agarwal. Frontierswe. *Proximal Blog*, 2026. https://frontierswe.com/blog.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Rishi Desai, Jesse Hu, Joan Cabezas, Neel Harsola, Pratyush Shukla, Roey Ben Chaim, Adnan El Assadi, Omkaar Mukund Kamath, Fenil Faldu, Prannay Hebbar, et al. Swe-marathon: Can agents autonomously complete ultra-long-horizon software work? *arXiv preprint arXiv:2606.07682*, 2026.

Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. A survey on the optimization of large language model-based agents. *ACM Computing Surveys*, 58 (9):1–37, 2026.

Yukang Feng, Jianwen Sun, Zelai Yang, Jiaxin Ai, Chuanhao Li, Zizhen Li, Fanrui Zhang, Kang He, Rui Ma, Jifan Lin, et al. Longcli-bench: A preliminary benchmark and study for long-horizon agentic programming in command-line interfaces. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 29952–29963, 2026.

Google DeepMind. Gemini 3.1 pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, 2026.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, 2026. URL https://doi.org/10.5281/zenodo.20953922.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. 2024. URL https://arxiv.org/abs/2403.07974.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Kimi. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long tasks, 2025. URL https://arxiv.org/abs/2503.14499.

Tue Le, Minh VT Thai, Dung Nguyen Manh, Huy Phan Nhat, and Nghi DQ Bui. Swe-evo: Benchmarking coding agents in long-horizon software evolution scenarios. *arXiv preprint arXiv:2512.18470*, 2025.

Zongxia Li, Wenhao Yu, Chengsong Huang, Zhenwen Liang, Rui Liu, Fuxiao Liu, Jingxi Che, Dian Yu, Jordan Boyd-Graber, Haitao Mi, et al. Self-rewarding vision-language model via reasoning decomposition. *arXiv preprint arXiv:2508.19652*, 2025.

Zongxia Li, Dawei Liu, Fuxiao Liu, Yuhang Zhou, Xiyang Wu, Jingxi Chen, Jing Xie, Xiaomin Wu, and Lichao Sun. Comfyclaw: Self-evolving skill harnesses for image generation workflows. *arXiv preprint arXiv:2607.01709*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Haojia Lin, Xiaoyu Tan, Yulei Qin, Zihan Xu, Yuchen Shi, Zongyi Li, Gang Li, Shaofei Cai, Siqi Cai, Chaoyou Fu, et al. Cuarewardbench: A benchmark for evaluating reward models on computer-using agent. *arXiv preprint arXiv:2510.18596*, 2025.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, et al. Harness updating is not harness benefit: Disentangling evolution capabilities in self-evolving llm agents. *arXiv preprint arXiv:2605.30621*, 2026.

Yue Liu, Yingwei Ma, Yibo Miao, Yanhao Li, Yuchong Xie, Xinlong Yang, Zhiyuan Hu, Flood Sung, Jiaheng Zhang, and Bryan Hooi. Klong: Training llm agent for extremely long-horizon tasks. *arXiv preprint arXiv:2602.17547*, 2026.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

MiniMax. Minimax sparse attention, 2026. URL https://arxiv.org/abs/2606.13392.

Moonshot AI. Kimi k2.6: From code to creation, from one to many. https://www.kimi.com/ai-models/kimi-k2-6, 2026. Official Moonshot AI model release page (Kimi K2.6).

Niels Mündler, Mark N Müller, Jingxuan He, and Martin Vechev. Swt-bench: Testing and validating real-world bug-fixes with code agents. *Advances in Neural Information Processing Systems*, 37: 81857–81887, 2024.

OpenAI. Codex. https://github.com/openai/codex, 2025a. OpenAI coding agent / CLI.

OpenAI. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/, 2025b. System card; also arXiv:2601.03267.

OpenAI. GPT-6 Astra system card. https://deploymentsafety.openai.com/gpt-6-astra/gpt-6-astra.pdf, September 2026. System card.

OpenClaw Contributors. Openclaw, 2026. URL https://github.com/openclaw/openclaw. Open-source agent platform.

Tianjun Pan, Xuan Lin, Wenyan Yang, Qianyu He, Shisong Chen, Licai Qi, Wanqing Xu, Hongwei Feng, Bo Xu, and Yanghua Xiao. Rubriceval: A rubric-level meta-evaluation benchmark for llm judges in instruction following. *arXiv preprint arXiv:2603.25133*, 2026.

Delip Rao and Chris Callison-Burch. Autorubric: Unifying rubric-based llm evaluation. *arXiv preprint arXiv:2603.00077*, 2026.

David Rein, Joel Becker, Amy Deng, Seraphina Nix, Chris Canal, Daniel O'Connel, Pip Arnott, Ryan Bloom, Thomas Broadley, Katharyn Garcia, et al. Hcast: Human-calibrated autonomy software tasks. *arXiv preprint arXiv:2503.17354*, 2025.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in llms. *arXiv preprint arXiv:2509.09677*, 2025.

Dingjie Song, Hanrong Zhang, Dawei Liu, Yixin Liu, Zongxia Li, Zhengqing Yuan, Siqi Zhang, and Lichao Sun. Dr. claw: An ai research workspace from idea to paper, 2026. URL https://github.com/OpenLAIR/dr-claw.

Zihang Tian et al. Arco: Adaptive rubric with co-evolution for multi-step llm-based agents, 2026. URL https://arxiv.org/abs/2606.21262.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, et al. Apex-agents. *arXiv preprint arXiv:2601.14242*, 2026.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents, 2024. URL https://arxiv.org/abs/2407.16741.

Xinyu Jessica Wang, Haoyue Bai, Yiyou Sun, Haorui Wang, Shuibai Zhang, Wenjie Hu, Mya Schroder, Bilge Mutlu, Dawn Song, and Robert D Nowak. The long-horizon task mirage? diagnosing where and why agentic systems break. *arXiv preprint arXiv:2604.11978*, 2026.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, et al. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024.

Xiyang Wu, Zongxia Li, Guangyao Shi, Alexander Duffy, Tyler Marques, Matthew Lyle Olson, Tianyi Zhou, and Dinesh Manocha. Co-evolving llm decision and skill bank agents for long-horizon tasks. *arXiv preprint arXiv:2604.20987*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL https://arxiv.org/abs/2404.07972.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024. URL https://arxiv.org/abs/2405.15793.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, et al. Harness-bench: Measuring harness effects across models in realistic agent workflows. *arXiv preprint arXiv:2605.27922*, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

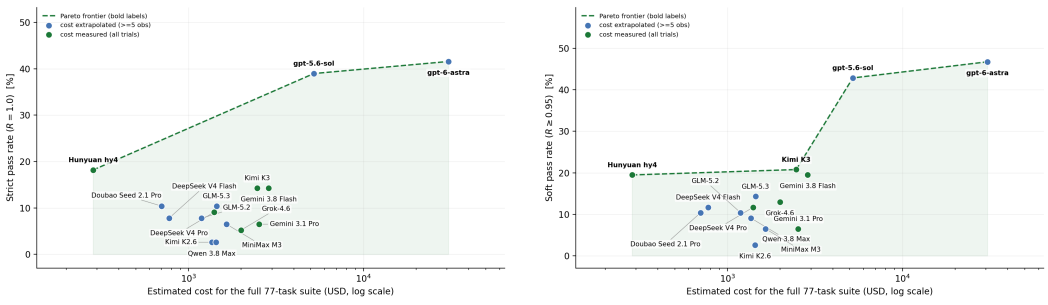
Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2023. URL https://arxiv.org/abs/2307.13854.

A Cost Analysis

We report the cost estimates reconstructed from per-task token usage and public list prices (Table 1), which Figure 7 summarizes with side-by-side Pareto cost–reward frontiers under strict and soft pass criteria. The estimated cost of running the full benchmark spans from about $285 for Hunyuan hy4 to about $30,581 for GPT-6-astra. Although GPT-6-astra achieves the highest pass rate, it does so at by far the highest total cost. GPT-5.6-sol forms a second tier of strong but substantially cheaper performance at roughly one sixth of that cost, while Kimi K3 sits further down the cost curve with much lower spend but also a noticeably lower resolution rate. Hunyuan hy4 anchors the low-cost end of the Pareto frontier, while GPT-5.6-sol and GPT-6-astra extend that frontier upward as progressively stronger but more expensive models. Under the soft $R \geq 0.95$ criterion, Kimi K3 joins the frontier because several of its near-misses are counted as passes rather than perfect completions. The remaining models form a middle or lower tier with moderate to high estimated costs but clearly lower resolution rates, indicating that higher inference spending alone is insufficient to guarantee better long-horizon performance.

Model	Tokens / task (M)	LLM calls / task	LLM time / task (min)	Cost / task ($)
Hunyuan hy4	4.4	78.9	102.7	3.71
Doubao Seed 2.1 Pro	18.0	200.2	111.7	9.12
DeepSeek V4 Flash	45.5	290.1	127.0	10.07
GLM-5.2	27.9	214.0	93.0	15.43
Qwen 3.8 Max	8.8	113.3	119.0	17.67
DeepSeek V4 Pro	31.2	231.0	130.8	18.21
Kimi K2.6	19.5	191.6	123.2	18.70
GLM-5.3	20.5	137.3	110.8	18.83
MiniMax M3	71.0	452.5	84.3	21.44
Grok-4.6	12.8	133.3	132.8	25.91
Kimi K3	16.2	170.8	93.7	32.09
Gemini 3.1 Pro	13.3	500.2	9.5	32.93
Gemini 3.8 Flash	48.2	836.4	12.7	37.31
GPT-5.6-sol	33.5	175.0	80.1	67.60
GPT-6-astra	39.4	213.5	83.2	397.16
Average	27.3	262.5	94.3	48.41

Table 1: **Estimated cost per model on LHTB 2.0.** On average, models spend 262.5 LLM calls and 94.3 minutes per task, at an estimated $48.4 per task. Costs are estimated from per-task token usage and public list prices. Tokens/task counts both input and output tokens, and Figure 7 uses the corresponding 77-task suite totals. † priced from the nearest listing rather than an exact match.



Strict pass rate ($R = 1.0$) **Soft pass rate** ($R \geq 0.95$)

Figure 7: **Cost–reward frontiers on LHTB 2.0.** Estimated cost for the full 77-task (log scale) against strict pass rate ($R = 1.0$, left) and soft pass rate ($R \geq 0.95$, right). The dashed line traces the Pareto frontier, and bold labels mark models on that frontier. Softening the pass criterion adds Kimi K3 to the frontier. Hy4 remains the best model at low cost rate while GPT-5.6-Astra is the best with the highest cost.

B Additional Failure and Domain Analyses

B.1 Dense Rewards Expose Different Failure Patterns

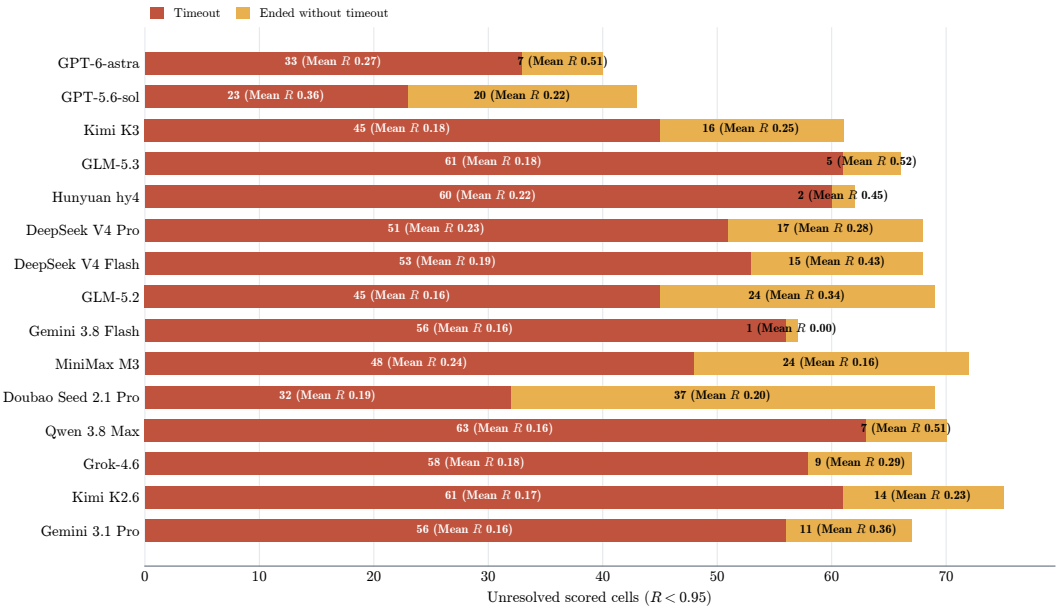


Figure 8: **Composition of unresolved scored cells per model.** Each bar decomposes a model's majorly unresolved scored cells ($R < 0.95$, out of 77) into timeouts (the agent is still working when the 3-hour budget expires) and endings without timeout. The label inside each timeout segment shows the number of timed-out cells and their mean reward; the label inside each non-timeout segment shows the number of endings and their mean reward. Timeouts account for 78% of all unresolved scored cells shown, indicating that many failures reflect incomplete long-horizon progress rather than immediate breakdown, while the smaller non-timeout segments highlight cases where agents stop despite not having fully solved the task.

Figure 8 shows unresolved scored cells by termination cause. Across all models, 78% of unresolved scored cells ($745/954$) end because the 3-hour budget expires while the agent is still actively working. However, these timed-out cells are generally not close to completion, with mean reward ranging from 0.16 to 0.36 across models. By contrast, endings without timeout account for only 22% of unresolved cells ($209/954$) and often indicate overconfidence: the agent stops on its own despite not yet satisfying the hidden verifier. Because the frozen export retains only the best scored cell for each model–task pair, infrastructure-only failures are not included in this figure.

The main bottleneck is not local execution correctness, but long-horizon completion. On Terminal Bench 2.0, most trials complete in under 20 minutes, and trajectory-level error analyses find that many failures are execution-related, including disobeying the specification, repeating steps, and missing termination conditions (Merrill et al., 2026). On LHTB 2.0, the more common failure is different: agents can often string together many locally correct actions, but they cannot reliably turn that progress into a finished artifact before the horizon expires. Short-horizon benchmarks primarily measure whether agents can *act correctly*; Long-Horizon-Terminal-Bench 2.0 additionally measures whether they can *budget a long horizon*, and the latter appears to be the binding constraint.

These results suggest that rankings on Long-Horizon-Terminal-Bench 2.0 reflect not only task-solving ability, but also *time efficiency*: how much reward a model can accumulate within a fixed budget. As a result, improvements that reduce redundant exploration, preserve state more effectively, or avoid repeated verification loops may yield larger gains than comparable improvements in single-step reasoning alone.

B.2 The Domains Are Genuinely Hard

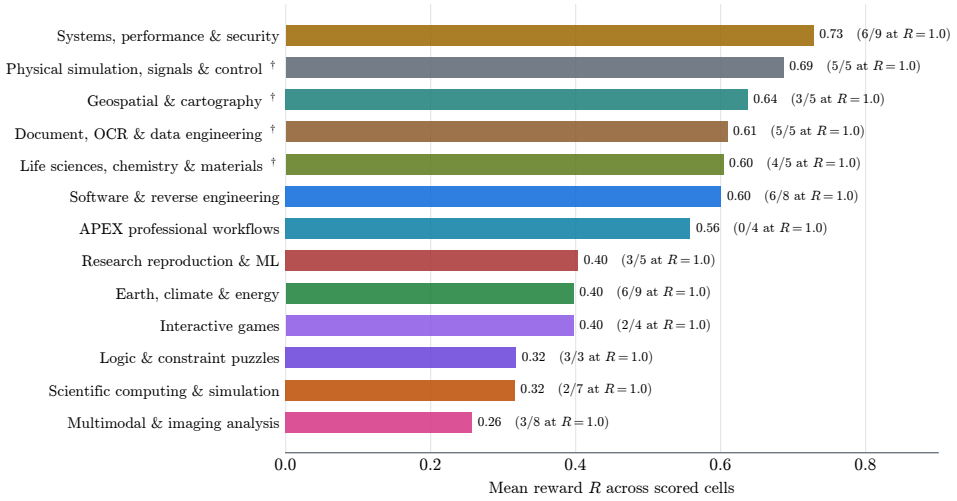


Figure 9: **Difficulty by task category on LHTB 2.0.** Each bar shows the dense mean reward R averaged over all scored cells in one of the 13 paper-level categories. The text at the right reports how many tasks in that category achieve a strict pass ($R = 1.0$) under at least one model.

Figure 9 shows substantial progress variation across the benchmark's 13 task categories. Multimodal & imaging analysis is the hardest category by mean reward ($R = 0.26$), followed by Scientific computing & simulation and Logic & constraint puzzles ($R = 0.32$ each). Systems, performance & security ($R = 0.73$), Physical simulation, signals & control ($R = 0.69$), and Geospatial & cartography ($R = 0.64$) are the most tractable. Mean reward and strict completion nevertheless capture different aspects of difficulty: APEX professional workflows has a mean reward of 0.56 but no strict passes, whereas all three Logic & constraint puzzle tasks are strictly solved by at least one model despite the category's low mean reward.

C List of Tasks in Long-Horizon-Terminal-Bench 2.0

Table 2 lists all 77 tasks in LHTB 2.0, grouped into 13 domains. Each row reports the task ID, category, a brief description, and a coarse difficulty label derived from mean reward across the evaluated models: tasks with mean reward at least 0.5 are labeled *Easy*, and the remainder are labeled *Hard*.

Table 2: **Task list for Long-Horizon-Terminal-Bench 2.0.** Each row shows a task ID, its paper-level category, a one-sentence description, and a coarse difficulty label derived from average task reward in Figure 3 (*Easy* if mean reward ≥ 0.5, otherwise *Hard*).

Task ID	Category	Description	Difficulty
2048	Interactive games	Play 2048 turn-by-turn via a terminal game server, merging tiles to reach the highest tile, scored by replaying recorded moves in a fresh seeded engine.	Hard
alp-paper-reproduction	Research reproduction & ML	Reproduce a calibrated ALP floating-point compression experiment from the paper's method and sample columns.	Hard
apex-ib244-matter	APEX professional workflows	Advise on taking EdTech firm KSchool private across 38 reactive stages, rebuilding LBO/DCF/comps models for exact figures.	Easy

Task ID	Category	Description	Difficulty
apex-investment-banking-matter	APEX professional workflows	Work a private-equity take-private of Planet Fitness through 19 reactive stages, rebuilding an LBO model and answering exact figures.	Hard
apex-law433-matter	APEX professional workflows	Act as outside counsel across four legal matters and 70 reactive stages over ~600 documents, submitting deterministically-graded answers.	Easy
apex-management-consulting-matter	APEX professional workflows	Work two management-consulting client matters across 33 reactive stages, rebuilding business-case and cost-savings models for exact answers.	Hard
apex-openroad-ibex-signoff	Software & reverse engineering	Close RTL-to-GDSII signoff on the ibex CPU with OpenROAD/sky130 across reactive stages until timing, DRC, LVS, area, and power pass.	Hard
audio-visual-event-alignment	Multimodal & imaging analysis	Repair an audio-video sync audit that detects visual events, audio onsets, dropped frames, and per-clip synchronization offsets.	Hard
chess-mate	Interactive games	Play chess as White against a bundled superhuman Stockfish engine, scored on how long the agent survives before being checkmated.	Hard
climate-netcdf-extreme-event-audit	Earth, climate & energy	Repair a climate audit pipeline computing gridded heatwave, precipitation, drought, anomaly, and index metrics over NetCDF datasets.	Hard
commit0-multilib-tdd	Software & reverse engineering	Reimplement three whole Python libraries from stubs with no provided tests, writing your own suite, graded on hidden real test suites.	Hard
dicom-radiology-audit	Multimodal & imaging analysis	Audit synthetic DICOM CT series geometry, HU conversion, windowing, masks, and lesion measurements.	Hard
document-table-layout-reconstruction	Multimodal & imaging analysis	Repair a scanned-document pipeline that detects table grids, assigns noisy OCR tokens to cells, and emits CSV/JSON/Markdown tables.	Hard
duckdb-optimizer-closure	Systems, performance & security	Iteratively patch DuckDB's query optimizer to speed up TPC-H while preserving correctness.	Hard
epa-swmm-stormwater-regression-audit	Earth, climate & energy	Audit EPA SWMM stormwater hydraulic and hydrologic simulations against official solver outputs.	Hard
epidemic-inverse-control-audit	Scientific computing & simulation	Fit and control a multi-city, age-structured SEIR-H-ICU epidemic model.	Hard
foldseek-paper-reproduction	Research reproduction & ML	Reproduce a calibrated Foldseek-style structural protein search experiment from the Nature Biotechnology paper.	Hard
gdal-proj-raster-regression	Earth, climate & energy	Reproduce GDAL/PROJ raster reprojection, coordinate transform, point sampling, and zonal-statistics regression checks.	Hard
generals-bot-arena	Software & reverse engineering	Develop a bot for a Generals.io-style grid strategy game by iteratively playing matches and refining its strategy.	Easy
grammar-fuzz-coverage-hunt	Systems, performance & security	Iteratively build a grammar-guided fuzzer to maximize code coverage of target parsers.	Easy
great-expectations-audit	Software & reverse engineering	Build a Great Expectations-based multi-table data-quality and reconciliation audit pipeline.	Hard
langchain-version-migration	Software & reverse engineering	Migrate a legacy LangChain RAG/router app across a breaking API change with hard-fail compatibility gates.	Hard
materials-phase-diagram-audit	Scientific computing & simulation	Repair a materials phase-diagram audit reconstructing phase boundaries, invariant reactions, lever-rule fractions, and convex-hull stability.	Hard

Task ID	Category	Description	Difficulty
matpower-opf-regression	Earth, climate & energy	Reproduce MATPOWER/PYPOWER power-flow and optimal-power-flow regression checks with grid-constraint diagnostics.	Hard
microscopy-cell-count-qc-audit	Multimodal & imaging analysis	Repair a microscopy QC audit that segments synthetic cell imagery, flags focus/artifact failures, and reports per-tile channel counts.	Hard
modflow6-groundwater-regression-audit	Earth, climate & energy	Audit USGS MODFLOW 6 groundwater simulations against official solver outputs.	Hard
nbody-accel-iterative	Scientific computing & simulation	Iteratively optimize an N-body gravitational simulation from O(N^2) to O(N log N) with correctness verification at multiple scales.	Easy
nrel-pysam-hybrid-renewables-audit	Earth, climate & energy	Audit renewable generation and storage dispatch against NREL PySAM official PVWatts outputs.	Hard
opensees-seismic-structural-regression-audit	Scientific computing & simulation	Audit seismic structural-dynamics results against official OpenSeesPy finite-element solver outputs.	Hard
poc-exploit-craft	Systems, performance & security	Generate proof-of-concept inputs that trigger specific vulnerability classes using sanitizer feedback.	Easy
riscv-core-debug	Software & reverse engineering	Localize and fix undisclosed injected bugs in an out-of-order RISC-V SystemVerilog CPU until simulation, compliance, and formal checks pass.	Easy
robotics-slam-benchmark-repair	Research reproduction & ML	Repair a SLAM benchmark audit that composes SE(2) poses, evaluates loop closures, reconstructs landmark maps, and reports trajectory consistency.	Hard
rush_hour_campaign	Logic & constraint puzzles	Solve a four-stage 6x6 Rush Hour campaign by hand, submitting legal move routes under length caps.	Hard
satellite-flood-change-detection-audit	Multimodal & imaging analysis	Repair a satellite flood change-detection audit that compares before/after imagery, masks clouds, and reports per-tile flood extent.	Hard
scientific-figure-data-reconstruction	Multimodal & imaging analysis	Repair a pipeline that reconstructs plotted series data from scientific figure images, captions, and partial CSV references.	Hard
snake_maze_campaign	Interactive games	Play a deterministic hard Snake variant, eating seeded apples and surviving growing obstacles as long as possible, scored by replaying the move log.	Hard
sokoban	Logic & constraint puzzles	Solve a ramp of Sokoban puzzles turn-by-turn through a game server, advancing as far as possible, scored by replaying moves in a clean engine.	Hard
spice-ephemeris-regression	Scientific computing & simulation	Reproduce NASA SPICE ephemeris, angle, and ground-station visibility regression checks from pinned kernels.	Hard
spot-scheduler-traces	Systems, performance & security	Implement and refine a cloud spot-instance scheduling policy that minimizes cost while meeting hard deadlines.	Easy
su2-airfoil-regression	Scientific computing & simulation	Reproduce a hardened SU2 aerodynamic regression matrix from official test-case references and solver history artifacts.	Hard
sudoku-recovery	Logic & constraint puzzles	Recover seven progressively corrupted hard variant-Sudoku puzzles turn-by-turn, detecting subtle corrupt givens via global constraint reasoning.	Hard
super-mario	Interactive games	Play Super Mario Bros turn-by-turn through a game server, clearing as many levels as possible, scored by replaying frame inputs in a fresh emulator.	Hard

Task ID	Category	Description	Difficulty
tabular-data-feature-covshift	Research reproduction & ML	Recover a sparse high-dimensional signal from low-dimensional observations under covariate shift, with a leakage-audited honesty gate.	Hard
unison-paper-reproduction	Research reproduction & ML	Reproduce a calibrated UNISON fat-tree network simulation experiment from the paper's method and settings.	Hard
unknown-config-semantics	Software & reverse engineering	Reverse-engineer an undocumented config format from a noisy spec and repair its normalization engine to pass hidden probes across five stages.	Hard
vector-db-iterative-build	Systems, performance & security	Build an approximate nearest-neighbor vector search service from scratch, iteratively optimizing recall and throughput.	Hard
acoustic_beamforming_peak__01	Physical simulation, signals & control	Reconstruct a room-acoustics beamforming pipeline so its peak-direction estimates match a reference simulator.	Hard
ajv_dynamic_ref_tree__05	Software & reverse engineering	Reconstruct AJV JSON-schema validation behaviour over dynamic $ref trees from observed accept/reject traces.	Hard
astro_mosaic_reprojection__04	Multimodal & imaging analysis	Reconstruct an astronomical FITS mosaic reprojection and photometry pipeline from reference outputs.	Hard
chem_reaction_smarts_application__02	Life sciences, chemistry & materials	Reconstruct SMARTS reaction-template application so product sets match a reference cheminformatics engine.	Hard
circuit_buck_converter_ripple__01	Physical simulation, signals & control	Reconstruct a buck-converter circuit simulation reproducing reference ripple and transient measurements.	Hard
climate_regrid_conservation_check__06	Earth, climate & energy	Reconstruct a NetCDF climate regridding pipeline that preserves the reference conservation diagnostics.	Hard
duckdb_recursive_hierarchy_report__03	Document, OCR & data engineering	Reconstruct a DuckDB recursive-hierarchy report matching reference aggregation output.	Hard
fea_plate_hole_stress_concentration__01	Physical simulation, signals & control	Reconstruct a finite-element plate-with-hole analysis reproducing reference stress-concentration factors.	Hard
font_harfbuzz_shaping_cluster__04	Document, OCR & data engineering	Reconstruct HarfBuzz text-shaping cluster and glyph-advance behaviour from observed shaping output.	Hard
gdal_antimeridian_clip__01	Geospatial & cartography	Reconstruct GDAL antimeridian-crossing clip geometry matching reference vector output.	Hard
gdal_vrt_mosaic_transform__03	Geospatial & cartography	Reconstruct a GDAL VRT mosaic and coordinate transform reproducing reference raster statistics.	Hard
genetic-convergence-testing	Scientific computing & simulation	See task instruction.	Hard
geology_multiwell_correlation_panel__04	Earth, climate & energy	Reconstruct a multi-well log correlation panel matching reference stratigraphic picks.	Hard
ghostscript_pdfa_preflight_repair__06	Document, OCR & data engineering	Reconstruct a Ghostscript PDF/A preflight-and-repair pipeline matching reference compliance reports.	Hard
hydro_hydrograph_response__06	Earth, climate & energy	Reconstruct a raster hydrology model reproducing reference hydrograph response curves.	Easy
lidar_icp_registration_summary__06	Geospatial & cartography	Reconstruct LiDAR ICP point-cloud registration reproducing reference alignment summaries.	Hard
map_label_priority_collision__04	Geospatial & cartography	Reconstruct cartographic label placement and collision priority matching reference rendered labels.	Hard

Task ID	Category	Description	Difficulty
map_road_class_generalization__05	Geospatial & cartography	Reconstruct road-class cartographic generalization matching reference simplified geometry.	Easy
materials_xrd_peak_shift__04	Life sciences, chemistry & materials	Reconstruct X-ray diffraction peak-shift analysis reproducing reference lattice measurements.	Hard
medical_slice_series_reassembly__03	Multimodal & imaging analysis	Reconstruct medical slice-series reassembly and geometry recovery matching reference volumes.	Hard
nginx_location_regex_tryfiles_precedence__06	Systems, performance & security	Reconstruct nginx location/regex/try_files routing precedence from observed request traces.	Hard
opa_rbac_namespace_boundary__05	Systems, performance & security	Reconstruct OPA/Rego RBAC namespace-boundary policy decisions from observed allow/deny traces.	Hard
openssl_pkits_name_constraints__05	Systems, performance & security	Reconstruct OpenSSL PKITS name-constraint certificate validation from reference verdicts.	Easy
pcap_http2_tls_fingerprint__04	Systems, performance & security	Reconstruct HTTP/2 and TLS fingerprint extraction from packet captures matching reference fingerprints.	Hard
pdb_multimodel_rmsd__02	Life sciences, chemistry & materials	Reconstruct multi-model PDB structural RMSD computation matching reference superposition values.	Hard
proteomics_retention_time_alignment__02	Life sciences, chemistry & materials	Reconstruct mass-spectrometry retention-time alignment matching reference peptide mappings.	Hard
robotics_pick_reach_controller__01	Physical simulation, signals & control	Reconstruct a robotic pick-and-reach controller reproducing reference trajectories and success criteria.	Easy
samtools_mpileup_consensus__05	Life sciences, chemistry & materials	Reconstruct a samtools mpileup consensus-calling pipeline matching reference variant output.	Hard
seismo_cross_correlation_lag__01	Physical simulation, signals & control	Reconstruct seismic waveform cross-correlation lag estimation matching reference arrival offsets.	Easy
sqlite_wal_recovery_audit__02	Document, OCR & data engineering	Reconstruct a SQLite WAL recovery audit reproducing reference post-crash database state.	Hard
tesseract_table_image_ocr__06	Document, OCR & data engineering	Reconstruct a Tesseract table-image OCR pipeline matching reference extracted tables.	Hard