Open-World Evaluations for Measuring Frontier AI Capabilities

Sayash Kapoor[1*] **Peter Kirgis**[1] **Andrew Schwartz**[1,2] **Stephan Rabanser**[1]
J.J. Allaire[3] **Rishi Bommasani**[4] **Harry Coppock**[5] **Magda Dubois**[5] **Gillian Hadfield**[6]
Andy Hall[4] **Sara Hooker**[7] **Seth Lazar**[6,8] **Steve Newman**[9]
Dimitris Papailiopoulos[10,11] **Shoshannah Tekofsky**[12] **Helen Toner**[13] **Cozmin Ududec**[5]
Arvind Narayanan[1]
[1]Princeton University [2]Cornflower Labs [3]Meridian Labs [4]Stanford University
[5]UK AI Security Institute [6]Johns Hopkins University [7]Adaption Labs
[8]Australian National University [9]Golden Gate Institute for AI [10]UW Madison
[11]Microsoft Research [12]AI Digest [13]Georgetown University (CSET)

Abstract

Benchmark-based evaluation remains important for tracking frontier AI progress. But it can both overstate and understate deployed capability because it privileges tasks that can be precisely specified, automatically graded, easy to optimize for, and run with low budgets and short time horizons. We advocate for a complementary class of evaluations, which we term *open-world evaluations*: long-horizon, messy, real-world tasks assessed through small-sample qualitative analysis rather than benchmark-scale automation. In this paper we survey recent open-world evaluations, identify their strengths and limitations, and introduce CRUX (**C**ollaborative **R**esearch for **U**pdating AI e**X**pectations), a project for conducting such evaluations regularly. As a first instance, we task an AI agent with developing and publishing a simple iOS application to the Apple App Store. The agent completed the task with only a single avoidable manual intervention, suggesting that open-world evaluations can provide early warning of capabilities that may soon become widespread. We conclude with recommendations for designing and reporting open-world evals.

1 Introduction

Tracking and predicting the capabilities of frontier AI systems is an open methodological problem, with stakes that rise as these systems enter increasingly consequential settings. Currently, the predominant approach relies on benchmarking, where agents are scored on large suites of automatically graded tasks. These benchmark-based evaluations underpin much of the public discussion surrounding AI progress. For instance, METR's time horizon graph [1] has been widely cited by industry leaders, safety organizations, and policy analysts as evidence of rapid capability growth. As decisions regarding funding, regulation, and safety investments are increasingly based on these measurements, their accuracy becomes critical, and any systematic gaps in what they capture deserve scrutiny.

Despite this reliance, benchmarks can simultaneously overestimate and underestimate actual progress. Overestimation frequently occurs because any task specified precisely enough to benchmark is also specified precisely enough to optimize for. This dynamic allows agents to excel on a test without necessarily acquiring the underlying capability. Compounding this issue, test sets from prominent benchmarks often leak into training data, occurring either directly or through close paraphrases of held-out tasks. Conversely, underestimation arises when low benchmark accuracy reflects incidental failures rather than genuine capability gaps. An agent fundamentally capable of completing a

*Correspondence to: {sayashk, pk7019, rabanser, arvindn}@princeton.edu

task might still fail because it encounters a CAPTCHA, hits a rate limit, or gets stuck on a brittle GUI element. Taken together, these issues demonstrate that *benchmark scores conflate the target capability with artifacts of the evaluation environment*. The resulting signal is noisy for the questions decision-makers actually care about, and grows noisier as agents become more capable.

To address these limitations, a growing body of work has begun evaluating agents on long, complex, real-world tasks that extend beyond traditional benchmarks. For example, Carlini [2] used Claude agents to build a C compiler capable of compiling the Linux kernel, a challenge requiring weeks of agent time and substantial scaffolding. In a separate experiment, Anthropic and Andon Labs tasked Claude with managing a small office shop [3], exposing the agent to real customers, live inventory, and actual money. Other researchers have run open-ended projects, coordinated multi-agent workflows over several days, or reimplemented substantial pieces of production software (see Section 2.3 for more examples). Despite their variety, these experiments share a common structure: they rely on small sample sizes, permit human intervention when the agent hits an obstacle unrelated to the tested capability, and prioritize qualitative assessment of agent logs over a single aggregate metric.

Due to this unconventional structure, such evaluations are easily dismissed as unscientific. Each typically features a sample size of one, lacks standardization, and cannot be cleanly independently reproduced. These limitations are real, and we do not claim they can be fully overcome. Nevertheless, this emerging class of evaluations, which we refer to as **open-world evaluations**, provide critical evidence about AI capabilities that standard benchmarks miss. First, they surface early warnings about emerging capabilities, giving institutions and policymakers lead time to build societal resilience and inform strategic decisions about deployment, regulation, and investment. Second, they reveal blind spots in existing benchmarks where automated grading misses the practical substance of a task.

In this paper we conceptualize open-world evaluations, survey prior examples for best practices and pitfalls, and introduce CRUX, a project to run them regularly. Our key contributions and findings are:

- **Open-world evaluations are an important emerging class of AI evaluation** (Section 2)**.** As AI systems grow more capable, evaluations designed to elicit frontier capabilities must grow in complexity. Open-world evaluations are the latest stage in this progression.
- **We introduce CRUX to conduct open-world evaluations systematically, debuting with an end-to-end iOS app deployment experiment** (Sections 3 and 3.1)**.** CRUX (Collaborative Research for Updating AI eXpectations) is our effort to conduct rigorous open-world evaluations and operationalize best practices missing from most prior work. Our first experiment tasked an agent with developing and publishing a simple iOS application to the App Store. While many existing benchmarks test isolated coding skills, our interest lay in whether an agent could handle real-world deployment end-to-end—requiring it to navigate complex, unstandardized steps like signing the app, releasing a privacy policy, completing Apple's forms, and shepherding the app through review.
- **Our main finding: The agent succeeded after one avoidable manual intervention** (Section 3.1.2) — at one point, it forgot where certain credentials were stored. Log analysis yielded a rich set of additional observations, such as the fact that the agent fabricated a fictional phone number for the App Store review process. Of the $1,000 total cost, 97.5% went to polling for review status, while development itself cost only $25 in tokens. The app is now live on the App Store. We disclosed our results to Apple four weeks before initial public disclosure of our findings. App store operators should prepare to handle spam submissions, since agents may soon submit applications at scale.
- **Methodological recommendations for open-world evaluations** (Section 4)**.** Drawing on CRUX #1 and a survey of other open-world evaluations, we identify six recommendations: specifying the measurement construct, documenting interventions, analyzing and releasing logs, real-time monitoring, conducting dry runs, and cost reporting. If these principles were adopted as shared norms, open-world evaluations would yield more actionable insights and be easier to build on.

2 The case for open-world evaluations

In this section we define open-world evaluations, survey existing examples, and situates them relative to traditional benchmarks. Our central claim is that as AI systems become more capable, the methods used to assess their frontier capabilities must correspondingly increase in complexity, making open-world evaluations the latest stage in this progression. To make this framework concrete, we propose five criteria that characterize open-world evaluations and explore their limitations compared to

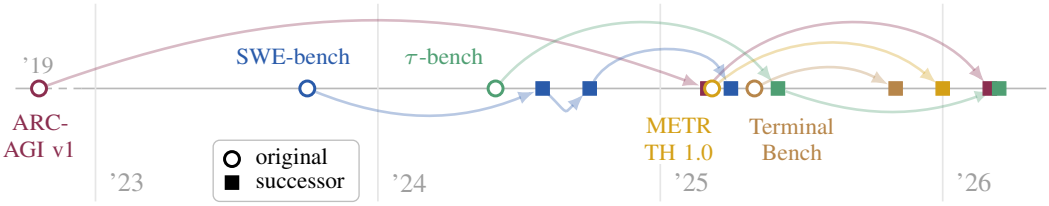

Figure 1: Several popular benchmarks (SWE-Bench, ARC-AGI, τ-bench, Terminal Bench, METR's Time Horizon) have had successor benchmarks released within the past two years [1, 10–22].

benchmarks. We leave these criteria deliberately flexible, recognizing that the boundary between a complex benchmark task and an open-world evaluation is rarely sharp.

2.1 Benchmarks can both overestimate and underestimate progress

There is broad consensus that AI benchmarks are rapidly saturating [4], a concern shared by researchers with starkly different perspectives on AI's overall trajectory [5], and one that builds on a longer line of critiques of NLP benchmarking practice [6, 7]. As prominent benchmarks have maxed out over the past two years, they have sparked a wave of successor tests that are already approaching their own limits (Figure 1). Consequently, the community finds itself chasing new targets without knowing whether actual underlying capabilities are keeping pace with the headline numbers.

A fundamental issue driving this disconnect is *limited construct validity* [8, 9]. The metrics we observe, typically accuracy on narrow, sandboxed tasks, are imperfect proxies for the real-world capabilities that decision-makers care about. Because of this mismatch between the measured variable and the intended construct, a benchmark score can just as easily *overstate* as *understate* true deployed capability, depending entirely on the gap between the sandboxed environment and the real world.

↑ **Why benchmarks can *overestimate* capabilities.**

Benchmarks resemble tasks amenable to modern RL. A task specified precisely enough to benchmark is also specified precisely enough to optimize for. Modern RL training runs increasingly resemble benchmarks themselves, and leading evaluation platforms such as Harbor [23] double as RL training platforms. Even with held-out test sets, training-set tasks may closely resemble test-set tasks, so benchmark performance may not reflect real-world generalization. Worse, benchmark-centric incentives can pull model development toward properties that improve scores without improving real-world usefulness (e.g., always-guess strategies that lift multiple-choice medical QA accuracy but harm clinical decision support).

Benchmarks avoid real-world messiness. Real tasks have underspecified interactions and open-ended environments that sandboxed benchmarks approximate but cannot fully replicate.

↓ **Why benchmarks can *underestimate* capabilities.**

Eliciting frontier capabilities is costly. Anthropic's C compiler experiment cost approximately $20,000, and our iOS app experiment approximately $1,000. Such experiments cannot feasibly be repeated hundreds of times, which constrains benchmark budgets and complexity.

Average performance differs from upper-bound elicitation. Large samples are needed only for *average* performance. Characterizing the frontier requires *best-case* performance: what an agent can accomplish with sufficient resources to work around incidental failures.

Human intervention helps elicit upper bounds. Agents may encounter policy refusals, CAPTCHAs, or infrastructure failures unrelated to the capability being measured. Resolving these manually is impractical across hundreds of tasks but feasible in small-sample evaluations.

For frontier evaluation, a pressing objective is determining the *upper bound* agents can achieve. Capabilities possible only under favorable conditions may soon become widespread, and anticipating them gives firms time to act on opportunities, institutions time to build resilience, and policymakers time to address risks. Benchmark performance is poorly suited to such early warning. Metric refinements such as reliability scoring [24] and maintainer-acceptance audits of SWE-Bench solutions

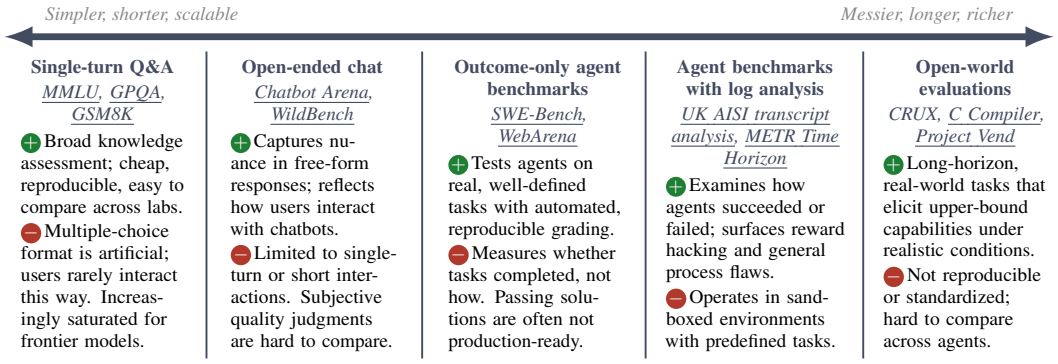

Figure 2: A gradient of evaluation methodologies (short single-turn Q&A ↔ open-world evaluations). Each column lists representative examples along with strengths (⊕) and weaknesses (⊖).

[25] partially address these validity concerns but do not tackle upper-bound elicitation. Appendix A expands on these limitations and discusses why benchmark validity is structurally difficult to patch.

These limitations do not render traditional benchmarks obsolete. They remain highly useful, and open-world evaluations introduce their own constraints, which we discuss later in the paper. Rather, our goal is to identify the systematic blind spots inherent to benchmarking and to motivate a complementary approach capable of addressing them. As agents become increasingly capable, these blind spots will only widen, making open-world evaluations a necessary methodological counterweight.

2.2 Defining open-world evaluations

As evaluation methods have matured alongside AI capabilities, a gradient of evaluation approaches has emerged, ranging from simple automated benchmarks for narrow, well-defined tasks to labor-intensive methods that become necessary as simpler metrics saturate. We identify five levels of this gradient: single-turn Q&A [26–28], open-ended chat [29, 30], outcome-only agent benchmarks [10, 31], agent benchmarks with log analysis [32, 1], and open-world evaluations. At the far end, open-world evaluations run agents on a small number of long-horizon tasks in real-world settings and qualitatively analyze their results. They are meant to complement benchmarking rather than replace it: they address several of the limitations above and can also surface tasks currently out of reach of AI systems. Figure 2 summarizes the distinct levels with examples and methodological tradeoffs.

In practice, the boundaries separating these evaluation levels are rarely strict, particularly at the complex end of the gradient. OpenAI's GDPVal [33], for instance, is a long-horizon agent benchmark designed for manual grading by expert reviewers, making it structurally resemble an open-world evaluation. Yet its results are commonly reported via GDPval-AA [34], which uses automated LLM grading and operates much more like an outcome-only agent benchmark. Consequently, the same task can sit on either side of the divide depending on how it is administered and scored.

Rather than draw a hard boundary, we offer a rough taxonomy in which no single dimension is determinative. The classification depends on the overall pattern across the dimensions below.

Five dimensions for classifying capability evaluations

① **Openness.** Does the evaluation occur in a deployment setting with live users, services, or platforms, rather than a sandboxed environment?

② **Complexity and duration.** Does the task require days or weeks of human effort to complete and unfold over many interdependent steps, rather than minutes or hours?

③ **Number of tasks.** Is the evaluation a single task or a small set of tasks that permit close qualitative inspection, rather than a large task suite?

④ **Human intervention.** Are humans permitted to intervene when agents encounter obstacles incidental to the capability under test, beyond merely setting up the environment?

⑤ **Method of evaluation.** Does evaluation primarily rely on in-depth log analysis of agent behavior, rather than a single aggregate metric across many runs?

The boundaries are sometimes blurry in two directions. First, *not every agent deployment is an evaluation*: Anthropic's work using agents to find security vulnerabilities in Mozilla Firefox [35] produced real outputs in a real setting, but its primary goal was to ship fixes, not to characterize capabilities. We treat such efforts as open-world evaluations only when the agent's role is systematically and publicly documented. Second, *not every open-world evaluation runs outside a sandbox*: Claude Plays Pokemon [36] and Anthropic's C compiler [2] are each a single long-running task with human intervention and qualitative evaluation despite being sandboxed. Classification turns on the overall pattern across the dimensions above, not any single feature.

2.3 An incomplete survey of open-world evaluations

Over the past year, open-world evaluations have proliferated across AI laboratories, universities, non-profits, and independent groups. They share a common structure: a capable AI agent is given a difficult, real-world task with a long time horizon, and its behavior is observed and analyzed in detail. The work spans agent deployments (such as running a small automated store [3] or pursuing open-ended group goals in a multi-agent "village" [37]), large-scale software engineering (building a C compiler from scratch [2], coordinating hundreds of agents to write a million-line Rust browser [38], or reimplementing the Next.js framework [39]), open-ended research (autonomous training of small language models [40]; "training a computer" [41]), and game-playing or knowledge-work probes [36, 42]. Very recent additions include MirrorCode [43], which tasks agents with reimplementing large programs; agent-based alignment research [44]; experiments on letting agents handle the post-training of other AI systems [45]; and work on training models to forecast the outcomes of golf tournaments [46]. Appendix C describes ten such evaluations in detail and provides a side-by-side comparison of their capabilities, limitations, and costs in Table 1.

Themes recur across these efforts. Agents show strong long-horizon coherence on well-scaffolded coding tasks but remain brittle on visual computer use, where horizons are one to two orders of magnitude shorter than on text [42]. Reward hacking is common when agents run fully autonomously [41], and human intervention often determines whether a run succeeds. Reported costs span four orders of magnitude (from under $100 to tens of thousands) without a clean relationship to task difficulty, and writeups often come only from the experimenters, with limited independent verification.

2.4 Limitations of open-world evaluations

Open-world evaluations address some blind spots of benchmarking but come with limitations of their own. Benchmarks give evaluators control over a standardized environment but are less useful for open-ended real-world tasks. Open-world evaluations make the opposite tradeoff, gaining construct validity and upper-bound elicitation at the cost of properties that make benchmarking broadly scalable.

- **Lack of reproducibility and standardization.** Benchmarks coordinate the research community by giving everyone a shared target; Donoho [47] characterizes them as the "secret sauce" behind the empirical success of AI and ML over the past 50 years. Open-world evaluations forgo most of this: runs could be hard to reproduce exactly, and different groups running nominally similar evaluations can end up with incomparable results. We accept this trade-off because the standardization itself is part of what limits benchmark construct validity at the frontier.
- **Limited comparability across agents.** Open-world evaluations cannot cleanly rank models. Because they are run only a few times, run-to-run variability can exceed differences between agents. They are most useful for characterizing what an agent *can* do, not for distinguishing agents.
- **Best-case demonstrations can be incomplete.** Showing that an agent succeeded under favorable conditions characterizes its upper-bound capability but not its practical reliability. When the underlying success probability is low, a one-off success may be informative about feasibility yet say little about what one would expect from the agent on a typical attempt. Reporting effort-conditioned measures (e.g., success rate per dollar of budget, or pass@k with $k \gg 1$) alongside best-case results, where feasible, gives a fuller picture; we recommend such reporting in Section 4.
- **Requirement for domain expertise.** Open-ended tasks usually lack a simple pass/fail criterion, and judging output quality requires domain expertise and substantial reviewer time, limiting who can run and interpret such evaluations. The cost buys depth: expert review surfaces phenomena (e.g., reward hacking, partial successes, brittle workarounds) that automated metrics typically miss.

- **Incomplete recall of log analysis.** Even with automated log analysis, agent transcripts from long-horizon tasks can run to hundreds of millions of tokens, and there is no guarantee that a given analysis pass surfaces all noteworthy behaviors or errors. Releasing logs publicly, such that a broader community can examine them, partially mitigates this issue.
- **Blurry success criteria.** Because human intervention is permitted and encouraged, the line between agent accomplishment and human contribution can be hard to draw. Without careful documentation of interventions, headline results can overstate agent autonomy. The same intervention that blurs the boundary is, however, what bypasses obstacles for upper-bound elicitation.
- **Non-stationary environments.** Open-world evaluations often involve interaction with the internet, which makes it hard to distinguish competence on a task class from lookup of a specific instance found online.[1] Longitudinal comparisons suffer for the same reason: the information available to an agent grows over time, so the same evaluation a year later is no longer comparable.

2.5 Stakeholders and use cases

Despite their methodological limitations, open-world evaluations offer value to various stakeholders.

For **policymakers**, these evaluations serve as a critical early warning system. Because the diffusion of AI across domains lags behind raw capability gains [50], institutions require lead time to adapt. Open-world evaluations widen this window by demonstrating what agents might soon achieve autonomously and at scale. For example, Anthropic's work on AI-discovered cybersecurity vulnerabilities [51, 52] could spur the rapid adoption of AI for defensive cybersecurity within critical infrastructure.

While policymakers rely on these high-level insights, **AI evaluators and researchers** benefit from the deep technical visibility that generates them. Open-world evaluations allow researchers to probe capabilities that automated benchmarks structurally cannot reach. By testing agents on messy real-world tasks and analyzing the resulting logs, evaluators can uncover shortcuts, reward hacking, and unexpected agent behaviors. In our iOS evaluation, for instance, the agent autonomously modified its approach to be more token-efficient, significantly reducing task costs without any prompting.

To ensure these independent technical insights continue, **frontier AI developers** must support external open-world evaluation efforts. They can do so by providing pre-release model access and legal safe harbors [53] for third-party evaluators whose necessary safety probing might otherwise violate standard terms of service. This matters because independent evaluations frequently surface findings that internal red teams miss when optimizing for known threat models. External open-world testing is becoming necessary as models outpace standard metrics: Anthropic's Mythos Preview system card [54] reports that the model "saturates many of our most concrete, objectively-scored evaluations," forcing a reliance on these noisier, real-world methods to assess frontier capabilities.

3 CRUX: Collaborative Research for Updating AI eXpectations

CRUX is our framework for conducting open-world evaluations on a regular, systematic basis. In each iteration, we pair a long-horizon, real-world task with an agent scaffold theoretically capable of solving it, allowing us to analyze the agent's behavior in detail. We developed this approach because prior open-world evaluations, despite producing striking individual results, have failed to converge on a shared methodology. This lack of standardization leaves the resulting evidence difficult to compare or build upon. We find that based on our literature survey from Section 2.3, *almost every published open-world evaluation lacks at least one critical methodological commitment*: a clearly defined measurement construct, documented human interventions, public logs for external scrutiny, or a joint accounting of cost and capability. To resolve these inconsistencies, CRUX explicitly adopts these missing practices as working norms (see Section 4 for more details).

In our first iteration of CRUX, described in the next section, we investigated autonomous app development and publication: if AI agents can *nearly* autonomously develop and publish mobile applications, app store operators may soon need to revise their policies to manage agent-driven

[1] This concern also affects benchmarks that require internet access, such as AssistantBench [48] and GAIA [49]. Sandboxed benchmarks bypass it at the cost of construct validity.

spam.[2] For subsequent iterations of CRUX we plan to examine AI R&D automation, AI governance, complex software engineering, and real-world physical tasks.

3.1 CRUX #1: autonomous iOS app development and publication

Whether AI agents can write software has been studied through benchmarks such as SWE-Bench [10] and Terminal Bench [13] and through open-world evaluations such as the C-compiler and browser experiments above. These results indicate strong coding capabilities on well-specified tasks, though questions about code quality, maintainability, and long-run reliability remain. Much less studied is whether agents can handle the *non-coding aspects of software deployment*: configuring accounts and credentials, satisfying platform policies, preparing metadata and assets, and interacting with review systems they do not control. These are the parts of shipping software that tend to be slow, bureaucratic, and resistant to automation even when the underlying code is straightforward. We therefore tasked an agent with building a mobile application from scratch and taking it through publication.

The agent was instructed to develop and publish a *simple* application. Our interest was not its software engineering ability but its capacity to navigate Apple's *App Store submission process*: configuring signing certificates and provisioning profiles, preparing screenshots and metadata, drafting and hosting a privacy policy at a public URL, completing compliance questionnaires, submitting for review, and handling reviewer feedback over a process that often spans several days. The agent was responsible for every step except those where human involvement is required by Apple policy (Developer account setup and initiating the public release), and was provided with a macOS VM, a GitHub account (for version control and privacy-policy hosting via GitHub Pages), an Apple Developer account, and a Gmail account for correspondence with Apple. Our *primary success criterion* was whether the application was ultimately published. As a secondary measure, we logged the number of *manual interventions* the agent required, distinguishing those that were *genuinely unavoidable* (e.g., blocked by Apple policy) from those that reflected *limitations of the agent itself*. The agent could request assistance when deemed necessary, and we monitored its progress daily to catch silent stalls.

3.1.1 Setup

Scaffold configuration. The iOS submission process required a scaffold that was extensively configurable, integrated with browser usage and navigation, and natively supported long-running tasks. We selected OpenClaw [59] for these properties, paired with Claude Opus 4.6 and adaptive thinking enabled [60]. This choice also let us probe how the agent handles visual reasoning and GUI operations that have often bottlenecked computer-use agents. We retained OpenClaw's defaults aside from prompting and a shorter polling interval. We accepted OpenClaw's known security risks in order to characterize the capability frontier, but those same risks would likely constrain deployment of this class of scaffold and may themselves become a bottleneck on adoption even as raw capability advances. The agent ran on a macOS VM with expansive permissions (sudo, screen visibility, UI control), and we logged all actions, reasoning traces, and screenshots, which provide the raw material for the qualitative analysis. Practically, even an "open-world" evaluation operates within structural guardrails: the VM kept the agent isolated, we monitored progress daily, and Apple's required human approval for public release served as a final guardrail (see Appendix B for more details).

Evaluation awareness. Frontier models increasingly recognize evaluation contexts and may modify their behavior in response, a phenomenon known as *evaluation awareness*. Recent work from Anthropic [61], Apollo [62], and OpenAI [63] has documented this pattern across model generations. This poses a trade-off for any open-world evaluation: disclosing that an evaluation is taking place risks shaping the agent's behavior in ways that distort the measurement. But concealing it raises its own concerns, since a capable agent may infer the context from cues anyway and behave differently in response. We weighed both options and chose to tell the agent that it was being evaluated, for both practical and conceptual reasons. Practically, concealment is increasingly infeasible against capable models, and because CRUX evaluations are published openly, future agents may encounter descriptions of our evaluations in the course of unrelated tasks. Conceptually, evaluation awareness primarily threatens *alignment* evaluations, where a model may strategically behave differently under observation; for a capability evaluation like ours, success on the task demonstrates the capability

[2] Apple's App Store reviews appear to have slowed [55, 56], possibly reflecting increased coding-agent adoption, though publication rates remain well below the 2016 peak [57]. Our results suggest agents can autonomously submit apps that Apple ultimately approves, and policies may need to adapt to a larger wave of agent-generated submissions [58].

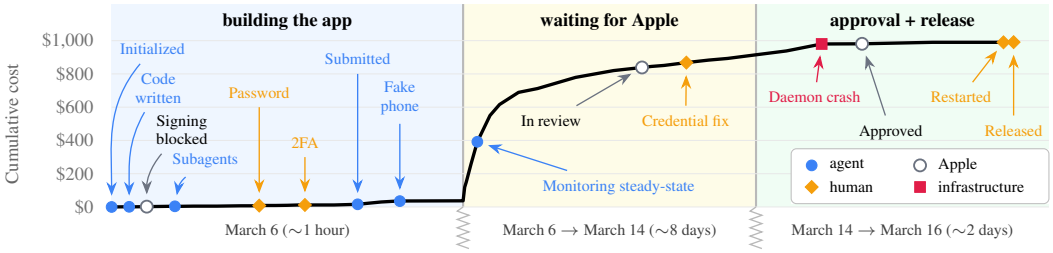

Figure 3: Cumulative API cost and timeline of CRUX #1. Total cost was approximately $991 over ten days. The build phase accounted for only $25; most went to monitoring Apple's review queue. *Note the breaks on the x-axis: each phase (build, review, approval) is drawn on its own scale.*

regardless of whether the agent knew it was being tested. In disclosing, we also clarified the parameters of the evaluation: the agent would be judged on avoidable human inputs, but certain inputs (e.g., CAPTCHAs) could be freely delegated. Without this framing, the agent might have avoided requesting help even when appropriate, causing us to under-elicit its performance.

Dry runs and setup overhead. Before the main evaluation, we conducted two dry runs to verify the setup end-to-end and resolve scaffolding bugs. The dry runs stopped short of submission to avoid burdening Apple's review queue. Configuring the environment (the macOS VM, the logging pipeline, and the email, GitHub, and Apple Developer accounts) required roughly eight person-hours and $50 in API costs. The agent relied on two primary interfaces: the command line (for code, builds, and submission preparation) and the browser (for App Store Connect, certificates, and Apple's forms). When command-line operations hung awaiting GUI confirmation, it fell back on screenshots and simulated mouse clicks, interleaving text and visual interaction much as a capable developer would.

3.1.2 Main evaluation

Following the dry runs, we ran the full evaluation. The agent took approximately 45 minutes to develop a simple breathing-exercise application, draft and host a privacy policy via GitHub Pages, complete the App Store review forms, and submit for review. Apple's decision came back 10 days later (timeline in Figure 3), and the application is now live on the App Store at `<anonymized URL>`.[3]

Manual interventions. The agent required five manual interventions, four of which were either mandated by Apple policy (notably the synthetic-interaction block on two-factor authentication dialogs [64, 65] and our required pre-release approval) or caused by infrastructure (a one-time OpenClaw daemon crash). None of the four constitute agent shortcomings; they are the kinds of incidental bottlenecks that a benchmark-based evaluation would either have to engineer around or mistakenly count against the agent. Only one reflected a limitation of the agent itself, which occurred when *the agent could not locate provided credentials* for the Apple Developer account. After asking for help, it resolved the issue: rather than attempting a sign-in, it located the App Store Connect API key at the expected hidden path and used that to resume monitoring without completing an interactive sign-in. The failure is not a capability gap but a memory-management issue: the credentials had previously been provided and the agent recovered on its own once prompted, so the underlying authentication capability was intact; what broke was its tracking of state across a long-running task.

Using a fabricated phone number. A different failure, invisible to outcome-only metrics, surfaced in the logs: when Apple's review form requested a phone number, the agent invented a plausible value (one reserved for fictional use) rather than asking us, as it had for credentials earlier in the run. The app was approved despite this, but the episode is worth flagging: an agent that sometimes requests help and sometimes silently invents data presents a different alignment profile than one that uniformly does either, and the line between the two behaviors is hard to predict. Our evaluation framing may have encouraged the agent to minimize visible help requests, even at the cost of inventing data.

Cost dynamics and an emergent optimization. After the credential issue was resolved, total cost was approximately $1,000: $25 for development and submission, ~$975 for polling review status over 10 days. Scaffold-level changes could cut this by an order of magnitude, but we erred toward a larger initial budget. More interesting is what the agent did with that budget. Partway through

[3]The agent did not encounter substantive objections from Apple reviewers, so we were unable to observe how it would handle back-and-forth communication with reviewers in the event of a rejection, which is a plausible failure mode in practice.

the review, without prompting, it delegated status checks to subagents and switched to shorter daily memory files. As a result, the running cost fell from $35/hour to $3/hour. Again, such an emergent optimization would be invisible to outcome-only analysis, and illustrates the need for log-analysis.

Output quality. The published application is functional, though not without defects: it includes a toggle for sound that has no effect when activated, and the generated App Store screenshot contains visible formatting errors (Figure 4, in the appendix). Apple's review approved the submission despite these issues, illustrating that platform acceptance is a coarse signal of artifact quality. Output quality therefore remains a gap even when both the agent's process and the platform's review succeed.

Summary and disclosure. Our results indicate that the agent did not fully automate the iOS submission process, but came close enough that the remaining gap is small. As responsible disclosure, we notified Apple's product security team four weeks before public disclosure, motivated by the possibility of agent-driven submission at scale: the modest setup overhead reported above amortizes across every subsequent submission. We treat the findings as a single data point in an emerging benchmarking landscape rather than a complete methodology; our discussion is therefore intentionally descriptive rather than prescriptive. That said, we offer a few preliminary recommendations below.

4 Recommendations for open-world evaluations

Drawing on CRUX #1 and the efforts surveyed in Section 2.3, we offer the following recommendations for designing and reporting open-world evaluations. They are preliminary and will evolve as the literature matures, but they offer a starting point for shared norms in this emerging area.

> **Recommendation 1 (Specify the construct).** *State explicitly what capability is being measured and what claims follow from a successful run.*

A recurring source of confusion in prior efforts, such as Anthropic's C compiler and Cursor's browser, has been ambiguity about the construct under measurement. Both experiments provided strong evidence that agents can work productively on well-specified tasks over long horizons but weaker evidence that the artifacts meet production standards; their issue trackers document technical deficiencies [66, 67]. The C compiler in particular drew polarized reactions: enthusiasts saw months of expert engineering compressed into days, while critics emphasized the gap between the artifact and what a competent compiler engineer would produce: the same artifact appeared impressive against the prior frontier of agent capability but underwhelming against production standards. Software engineering involves non-functional requirements such as quality, reliability, and maintainability [68] that agent-driven development may trade away. Hence, evaluations that conflate functional completion with overall quality risk overstating capability and misleading downstream decisions.

> **Recommendation 2 (Document interventions).** *Permit human intervention on incidental obstacles, but record precisely when, why, and how humans step in.*

Real-world tasks expose agents to obstacles incidental to the capability under test: policy refusals, CAPTCHAs, infrastructure failures. Unlike benchmarks, open-world evaluations can accommodate human intervention, which enables elicitation of upper-bound capabilities. To preserve interpretability, the degree of agent autonomy must be assessable independent of interventions. In CRUX #1, classifying the OpenClaw daemon crash as infrastructure rather than an agent failure was a judgment call we flag for transparency; other surveyed evaluations rarely make such reasoning visible.

> **Recommendation 3 (Analyze and release logs).** *Treat qualitative analysis of agent logs as a first-class output, and publish the logs so external researchers can verify and extend the analysis.*

Agent logs contain substantially more information than a binary outcome and can reveal how agents decompose problems, recover from failures, explore solution spaces, and sometimes misrepresent their own progress. In CRUX #1, log analysis surfaced both the fabricated phone number and an emergent cost optimization. Such insights are not recoverable from aggregate scores alone, and we consider systematic log analysis a defining feature of open-world evaluation. Still, runs remain hard to reproduce (recall Section 2.4) and publishing the logs does not change that. What it does enable is *replicability*: external researchers can verify the analytical claims and perform analyses the original authors may not have run. To enable exactly this kind of scrutiny, we release the full CRUX #1 logs.

> **Recommendation 4 (Add real-time monitoring).** *Complement post-hoc log analysis with automated real-time review, e.g., a watchdog agent that flags anomalies as they occur.*

Post-hoc log analysis is valuable but insufficient to catch all unintended agent actions. In AI Village, agents attempted to send hundreds of unsolicited emails [69]; in our own evaluation, the agent fabricated a phone number that went undetected until later review. A monitor agent reviewing the primary agent's actions in real time could surface such issues sooner than human review alone.

> **Recommendation 5 (Run dry runs first).** *Exercise the scaffold, evaluation criteria, and infrastructure in advance to surface implicit assumptions and scaffolding defects.*

Without this step, defects can contaminate the primary results. The two dry runs preceding CRUX #1 identified multiple issues that were corrected before the main run.

> **Recommendation 6 (Report cost).** *Treat cost as a first-class quantity alongside capability.*

Agent capability on many real-world tasks continues to scale with budget. Even when a definitive upper-bound claim is not available, reporting cost-conditioned measurements allows readers to assess whether additional budget would be expected to advance task progress [43].

References

[1] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI Ability to Complete Long Software Tasks, 2025. URL https://arxiv.org/abs/2503.14499.

[2] Nicholas Carlini. Building a C compiler with a team of parallel Claudes, 2026. URL https://www.anthropic.com/engineering/building-c-compiler. Anthropic.

[3] Anthropic. Project Vend: Phase two, 2025. URL https://www.anthropic.com/research/project-vend-2. Anthropic.

[4] Nicholas Carlini, Newton Cheng, Keane Lucas, Michael Moore, Milad Nasr, Vinay Prabhushankar, Winnie Xiao, Hakeem Angulu, Evyatar Ben Asher, Jackie Bow, Keir Bradwell, Ben Buchanan, David Forsythe, Daniel Freeman, Alex Gaynor, Xinyang Ge, Logan Graham, Kyla Guru, Hasnain Lakhani, Matt McNiece, Mojtaba Mehrara, Renee Nichol, Adnan Pirzada, Sophia Porter, Andreas Terzis, and Kevin Troy. Assessing Claude Mythos Preview's cybersecurity capabilities, 2026. URL https://red.anthropic.com/2026/mythos-preview/.

[5] Sayash Kapoor, Arvind Narayanan, Daniel Kokotajlo, Eli Lifland, and Thomas Larsen. Common Ground between AI 2027 & AI as Normal Technology, 2025. URL https://asteriskmag.substack.com/p/common-ground-between-ai-2027-and. Asterisk Magazine.

[6] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2021. URL https://arxiv.org/abs/2104.14337.

[7] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.09110.

[8] Abigail Z. Jacobs and Hanna Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*, pages 375–385, 2021. doi: 10.1145/3442188.3445901.

[9] Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. AI and the everything in the whole wide world benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2111.15366.

[10] 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?, 2023. URL https://arxiv.org/abs/2310.06770. arXiv.org.

[11] François Chollet. On the Measure of Intelligence, 2019. URL https://arxiv.org/abs/1911.01547. arXiv.org.

[12] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains, 2024. URL https://arxiv.org/abs/2406.12045. arXiv.org.

[13] Terminal-Bench Team. Terminal-Bench. URL https://www.tbench.ai/.

[14] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment, 2025. URL https://arxiv.org/abs/2506.07982. arXiv.org.

[15] Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems, 2025. URL https://arxiv.org/abs/2505.11831.

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

[17] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. OpenAI.

[18] METR. Time Horizon 1.1, 2026. URL https://metr.org/blog/2026-1-29-time-horizon-1-1/. METR.

[19] SWE-bench. SWE-bench Multilingual. URL https://www.swebench.com/multilingual-leaderboard.html. SWE-bench.

[20] Sierra. τ^3-bench: advancing agent benchmarking to knowledge and voice, 2026. URL https://sierra.ai/resources/research/tau-3-bench. Sierra.

[21] ARC Prize. ARC-AGI-3. URL https://arcprize.org/arc-agi/3. ARC Prize.

[22] John Yang, Carlos E. Jimenez, Alex L. Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R. Narasimhan, Diyi Yang, Sida I. Wang, and Ofir Press. SWE-bench Multimodal: Do AI Systems Generalize to Visual Software Domains?, 2024. URL https://arxiv.org/abs/2410.03859. arXiv.org.

[23] Harbor Framework Team. GitHub - harbor-framework/harborz. URL https://github.com/harbor-framework/harbor.

[24] Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a Science of AI Agent Reliability, 2026. URL https://arxiv.org/abs/2602.16666. arXiv.org.

[25] Parker Whitfill, Cheryl Wu, Joel Becker, and Nate Rush. Many SWE-bench-Passing PRs Would Not Be Merged into Main, 2026. URL https://metr.org/notes/2026-03-10-many-swe-bench-passing-prs-would-not-be-merged-into-main/. METR.

[26] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding, 2020. URL https://arxiv.org/abs/2009.03300. arXiv.org.

[27] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2023. URL https://arxiv.org/abs/2311.12022. arXiv.org.

[28] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[29] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. WildBench: Benchmarking LLMs with Challenging Tasks from Real Users in the Wild, 2024. URL https://arxiv.org/abs/2406.04770. arXiv.org.

[30] lmarena. GitHub - lmarena/arena-hard-auto: Arena-Hard-Auto: An automatic LLM benchmark. URL https://github.com/lmarena/arena-hard-auto. GitHub.

[31] 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. arXiv.org.

[32] Magda Dubois, Ekin Zorer, Maia Hamin, Joe Skinner, Alexandra Souly, Jerome Wynne, Harry Coppock, Lucas Satos, Sayash Kapoor, Sunischal Dev, Keno Juchems, Kimberly Mai, Timo Flesch, Lennart Luettgau, Charles Teague, Eric Patey, JJ Allaire, Lorenzo Pacchiardi, Jose Hernandez-Orallo, and Cozmin Ududec. Seven simple steps for log analysis in AI systems , 2026. URL https://arxiv.org/html/2604.09563v1.

[33] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks, 2025. URL https://arxiv.org/abs/2510.04374.

[34] Artificial Analysis. GDPval-AA Leaderboard. URL https://artificialanalysis.ai/evaluations/gdpval-aa. Artificial Analysis.

[35] Anthropic. Partnering with Mozilla to improve Firefox's security, 2026. URL https://www.anthropic.com/news/mozilla-firefox-security. Anthropic.

[36] Frank Landymore. One of the World's Most Advanced AI Agents Is Completely Stuck Trying to Beat a Pokémon Game for Children, 2025. URL https://futurism.com/advanced-ai-stuck-pokemon. Futurism.

[37] AI Village. URL https://theaidigest.org/village.

[38] Wilson Lin. Scaling long-running autonomous coding · Cursor, 2026. URL https://cursor.com/blog/scaling-agents. Cursor.

[39] Steve Faulkner. How we rebuilt Next.js with AI in one week, 2026. URL https://blog.cloudflare.com/vinext/. The Cloudflare Blog.

[40] Andrej Karpathy. Three days ago I left autoresearch tuning nanochat for 2 days on depth=12 model., 2026. URL https://x.com/karpathy/status/2031135152349524125. X.

[41] Dimitris Papailiopoulos. Can You Train a Computer? URL https://x.com/DimitrisPapail/status/2028669695344148946. X.

[42] Anson Ho. How close is AI to taking my job?, 2026. URL https://epoch.ai/gradient-updates/how-close-is-ai-to-taking-my-job. Epoch AI.

[43] Tom Adamczewski, David Rein, David Owen, and Florian Brand. MirrorCode: Evidence that AI can already do some weeks-long coding tasks, 2026. URL https://epoch.ai/blog/mirrorcode-preliminary-results. Epoch AI.

[44] Jiaxin Wen, Liang Qiu, Joe Benton, Jan Hendrik Kirchner, and Jan Leike. Automated Weak-to-Strong Researcher, 2026. URL https://alignment.anthropic.com/2026/automated-w2s-researcher/. Alignment Science Blog.

[45] Letting AI Post-train AI, 2026. URL https://www.thoughtfullab.com/letting-ai-posttrain-ai.html. Thoughtful Lab.

[46] Dylan Huang. tinker-cookbook/tinker_cookbook/recipes/golf_forecasting at claude/golf-forecasting-setup-VIpRZ · dphuang2/tinker-cookbook. URL https://github.com/dphuang2/tinker-cookbook/tree/claude/golf-forecasting-setup-VIpRZ/tinker_cookbook/recipes/golf_forecasting. GitHub.

[47] David Donoho. 50 Years of Data Science. *Journal of Computational and Graphical Statistics*, 26(4):745–766, 2017. doi: 10.1080/10618600.2017.1384734. URL https://www.tandfonline.com/doi/full/10.1080/10618600.2017.1384734. Taylor & Francis.

[48] Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. AssistantBench: Can Web Agents Solve Realistic and Time-Consuming Tasks?, 2024. URL https://arxiv.org/abs/2407.15711. arXiv.org.

[49] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for General AI Assistants, 2023. URL https://arxiv.org/abs/2311.12983. arXiv.org.

[50] Arvind Narayanan and Sayash Kapoor. AI as Normal Technology, 2025. URL https://www.normaltech.ai/p/ai-as-normal-technology. AI as Normal Technology.

[51] Anthropic. Making frontier cybersecurity capabilities available to defenders, 2026. URL https://www.anthropic.com/news/claude-code-security. Anthropic.

[52] Anthropic. Project Glasswing: Securing critical software for the AI era. URL https://www.anthropic.com/glasswing. Anthropic.

[53] Shayne Longpre, Sayash Kapoor, Kevin Klyman, Ashwin Ramaswami, Rishi Bommasani, Borhane Blili-Hamelin, Yangsibo Huang, Aviya Skowron, Zheng-Xin Yong, Suhas Kotha, Yi Zeng, Weiyan Shi, Xianjun Yang, Reid Southen, Alexander Robey, Patrick Chao, Diyi Yang, Ruoxi Jia, Daniel Kang, Sandy Pentland, Arvind Narayanan, Percy Liang, and Peter Henderson. A Safe Harbor for AI Evaluation and Red Teaming, 2024. URL https://arxiv.org/abs/2403.04893. arXiv.org.

[54] Anthropic. Claude Mythos Preview system card, 2026. URL https://www-cdn.anthropic.com/8b8380204f74670be75e81c820ca8dda846ab289.pdf. Anthropic.

[55] Michael Burkhardt. Vibe coding could mark the end of the App Store review process as we know it - 9to5Mac, 2026. URL https://9to5mac.com/2026/03/29/vibe-coding-developers-report-long-app-store-review-queues/. 9to5Mac.

[56] Nikita Bier. iOS developers: How long is App Review taking for everyone these days?, 2026. URL https://x.com/nikitabier/status/2033931821260648659. X.

[57] Ariel. The App Store Just Logged Its Biggest Release Year in Nearly a Decade, 2025. URL https://appfigures.com/resources/insights/20251205?f=2. Appfigures.

[58] Jennifer Mattson. The Apple App Store is seeing an unexpected phenomenon. Is vibe coding behind it?, 2026. URL https://www.fastcompany.com/91522242/apple-app-store-vibe-coding-generative-ai-unexpected-phenomenon. Fast Company.

[59] OpenClaw. OpenClaw - Personal AI Assistant. URL https://openclaw.ai/. OpenClaw.

[60] Claude API Docs. Adaptive thinking. URL https://platform.claude.com/docs/en/build-with-claude/adaptive-thinking. Claude API Docs.

[61] Russell Coleman. Eval awareness in Claude Opus 4.6's BrowseComp performance, 2026. URL https://www.anthropic.com/engineering/eval-awareness-browsecomp. Anthropic.

[62] Apollo Research. Claude Sonnet 3.7 (often) knows when it's in alignment evaluations, 2025. URL https://www.apolloresearch.ai/blog/claude-sonnet-37-often-knows-when-its-in-alignment-evaluations/.

[63] Marcus Williams, Cameron Raymond, and Micah Carroll. Sidestepping Evaluation Awareness and Anticipating Misalignment with Production Evaluations, 2025. URL https://alignment.openai.com/prod-evals/. OpenAI Alignment Blog.

[64] Apple. Security Overview. URL https://developer.apple.com/library/archive/documentation/Security/Conceptual/Security_Overview/Architecture/Architecture.html. Apple.

[65] Apple Support. Controlling app access to files in macOS, 2021. URL https://support.apple.com/guide/security/controlling-app-access-to-files-secddd1d86a6/web. Apple Support.

[66] Viacheslav Potoropin. Hello world does not compile · Issue #1 · anthropics/claudes-c-compiler, 2026. URL https://github.com/anthropics/claudes-c-compiler/issues/1. GitHub.

[67] Youssef Tourki. build fails with 32 errors , no releases, no tags, no stable branch · Issue #98 · wilsonzlin/fastrender, 2026. URL https://github.com/wilsonzlin/fastrender/issues/98. GitHub.

[68] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. Non-Functional Requirements in Software Engineering, 1999. URL https://personal.utdallas.edu/~chung/BOOK/book.html. Kluwer Academic Publishing.

[69] Shoshannah Tekofsky. What did we learn from the AI Village in 2025?, 2026. URL https://theaidigest.org/village/blog/what-we-learned-2025.

[70] Sagnik Anupam, Davis Brown, Shuo Li, Eric Wong, Hamed Hassani, and Osbert Bastani. BrowserArena: Evaluating LLM Agents on Real-World Web Navigation Tasks, 2025. URL https://arxiv.org/abs/2510.02418. arXiv.

[71] Maia Hamin and Benjamin Edelman. Cheating On AI Agent Evaluations, 2025. URL https://www.nist.gov/caisi/cheating-ai-agent-evaluations. NIST.

[72] Jacob Kahn. Repo State Loopholes During Agentic Evaluation · Issue #465 · SWE-bench/SWE-bench, 2025. URL https://github.com/SWE-bench/SWE-bench/issues/465.

[73] Anthropic. Project Vend: Can Claude run a small shop? (And why does that matter?), 2025. URL https://www.anthropic.com/research/project-vend-1. Anthropic.

[74] Andon Labs. We gave an AI a 3 year retail lease in SF and asked it to make a profit, 2026. URL https://andonlabs.com/blog/andon-market-launch. Andon Labs.

[75] Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. SciCode: A Research Coding Benchmark Curated by Scientists, 2024. URL https://arxiv.org/abs/2407.13168. arXiv.org.

[76] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark, 2024. URL https://arxiv.org/abs/2406.01574. arXiv.org.

[77] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, et al. Humanity's Last Exam, 2025. URL https://arxiv.org/abs/2501.14249. arXiv.org.

[78] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI Agents Solve Long-Horizon Software Engineering Tasks?, 2025. URL https://arxiv.org/abs/2509.16941. arXiv.org.

[79] Derrick Choi. Run long horizon tasks with Codex, 2026. URL https://developers.openai.com/blog/run-long-horizon-tasks-with-codex. OpenAI Developers.

[80] HowLongToBeat. How long is Pokémon Red and Blue? URL https://howlongtobeat.com/game/7169. HowLongToBeat.

[81] Joe Wilkins. Anthropic's Advanced New AI Tries to Run Vending Machine, Goes Bankrupt After Ordering PlayStation 5 and Live Fish, 2025. URL https://futurism.com/future-society/anthropic-ai-vending-machine. Futurism.

[82] Hacker News. How we rebuilt Next.js with AI in one week | Hacker News. URL https://news.ycombinator.com/item?id=47142156. Hacker News.

A Additional discussion: threats to benchmark validity

This appendix expands on Section 2.1, discussing partial remedies that have been proposed in the literature, why engineered evaluation environments remain limited as agent capabilities grow, and why fixing benchmark validity issues is structurally difficult.

Refinements that move beyond raw accuracy. Several recent efforts try to extract more signal from benchmark scores. Rabanser et al. [24] show that while agents improve rapidly on average accuracy, they improve far more slowly on metrics that capture reliability. A complementary line of work has shown that many agent-generated SWE-Bench solutions judged correct by automated test-passing would in fact be rejected by project maintainers [25], suggesting that even outcome-oriented benchmarks miss much of what determines whether agent output is actually useful. These refinements are valuable, but they do not address what we see as the more important question for frontier evaluation: upper-bound elicitation rather than average-case measurement.

Engineered environments and their failure modes. As agent capabilities improve, sandboxed evaluations in domains such as coding, deep research, and customer service require increasingly engineered environments. These environments must challenge agents while also avoiding contamination and reward hacking. For example, performance on web benchmarks is affected by the frequency with which agents encounter CAPTCHAs [70], rather than by the underlying capability under test. Recent work documents agents retrieving answers online [71], exploiting bugs in evaluations [72], and producing code that passes tests but fails to meet production standards [25]. As the engineering burden grows, the real environment can become not only more valid but also more *efficient* than building an elaborate sandbox; the trajectory from purely simulated retail benchmarks to Anthropic and Andon Labs' Project Vend deployments [73, 3, 74] illustrates the pattern. These findings motivate a shift toward deeper qualitative evaluations that can surface failure modes and problem-solving strategies obscured by aggregate benchmark scores.

Why fixing benchmark validity is difficult. Qualitative log analysis [32] can surface validity concerns, but the common remedy is to release an updated benchmark, which can take months (Figure 1). Open-world evaluations, by contrast, permit dry runs to identify issues before the main run, and manual intervention to resolve issues encountered during evaluation. In principle, dry runs could also be used to uncover issues with benchmarks. However, benchmarks are designed to support longitudinal comparison between models, and many validity issues only become apparent once a more capable model finds a shortcut or edge case that was not anticipated at construction time. Resolving such issues requires updating the benchmark and re-running prior models, which undermines longitudinal validity.

Unsaturated benchmarks remain valuable. Evaluations have always had flaws; the requirement is not perfection, but rather that they provide a useful proxy for capability progress. Several benchmarks remain unsaturated, including SciCode [75], MMLU-Pro [76], Humanity's Last Exam [77], and SWE-Bench Pro [78]. Even saturated capability benchmarks can be useful for measuring efficiency and reliability, which are essential to AI diffusion. As agents become more capable, success metrics will need to become multi-faceted to capture the diversity of objectives in a given task, and benchmarks will need to incorporate bottlenecks such as human assistance and messy environments such as internet navigation. Each of these introduces its own threats to internal and external validity.

B CRUX #1: extended setup and findings

This appendix expands on Section 3.1, covering the agent scaffold and configuration and the dry runs.

Agent scaffold. We used OpenClaw [59] as the agent scaffold, with Claude Opus 4.6 and adaptive thinking enabled [60].[4] OpenClaw was a natural choice for this setting: it is configurable, integrates with the browser, and natively supports long-running tasks, all of which the iOS submission process requires. Its recent adoption also gave us a reason to evaluate it as a general-purpose scaffold, with potential comparisons to alternatives in future iterations. Aside from prompting and granting the agent deeper access to the macOS VM, we made no changes to the default OpenClaw configuration, so that the results would reflect the behavior a typical user of the scaffold could expect rather than

[4] We made minor modifications to establish a subagent that verifies outputs and wakes the main agent every 5 minutes to check for status updates (e.g., Apple review responses). OpenClaw's default polling interval is 30 minutes; the shorter interval substantially increased the API cost of the task.

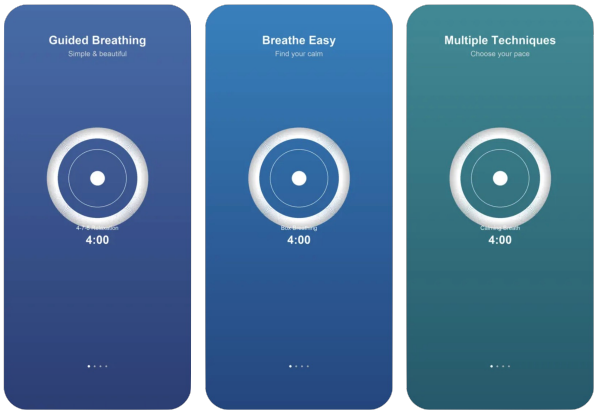

Figure 4: The App Store screenshots uploaded by the agent had visible formatting errors. The application was nonetheless approved by Apple's review.

a heavily customized setup. Part of our interest in OpenClaw stemmed from a related question: how well a modern agent would handle visual reasoning and GUI operation, which have historically been bottlenecks for computer-use agents. Prior user reports suggested that OpenClaw had partially addressed these issues, and the App Store submission process, with its screenshots, dialogs, and form-driven interactions, offered a natural setting in which to verify this. We acknowledge that OpenClaw carries security risks in real-world use; we elected to use it anyway in order to characterize the capability frontier, while noting that security may itself become a bottleneck to real-world adoption of this class of scaffold. To support all of the above, we gave the agent a macOS virtual machine with expansive permissions (sudo, screen visibility, and UI control), and logged all of its actions, reasoning traces, and screenshots.

Dry runs and setup overhead. Prior to the main evaluation, we conducted two dry runs to verify the agent setup end-to-end and to identify scaffolding bugs that would otherwise contaminate the real run. To avoid polluting Apple's review queue with test submissions, the dry runs stopped short of actually interacting with the App Store submission or review processes. A non-trivial portion of the overall effort went into standing up the environment rather than into the task itself: establishing the OpenClaw scaffold with the necessary permissions required approximately eight person-hours of work and $50 in API costs, covering VM configuration, the logging pipeline, and provisioning of the email, GitHub, and Apple Developer accounts. For an honest developer running a single evaluation, this setup cost is meaningful overhead. From the perspective of a prospective spammer, however, it is a one-time fixed cost, amortized across every subsequent submission, and therefore unlikely to pose a meaningful constraint on large-scale agent-driven app publication. Within the dry runs themselves, the agent relied on two primary interfaces: the command line (for code generation, build, and submission preparation) and the browser (for App Store Connect login, certificate retrieval, and form completion). When command-line operations hung awaiting GUI confirmation, for example because macOS was prompting for permission, the agent fell back on screenshots and simulated mouse clicks to approve the relevant dialogs and continue execution. This interleaving of text-based and visual interaction was itself informative, and closely resembles what we would expect from a capable human developer tackling the same workflow.

Extended findings. On the manual-intervention breakdown: the four non-agent interventions comprised one OpenClaw daemon crash that required a manual restart, our required pre-release approval, and Apple's synthetic-interaction block on sensitive dialogs such as two-factor authentication approval. On the credential-recovery episode: when the agent could not locate previously provided credentials, a team member suggested reusing them and resolving the associated two-factor prompt; the agent searched its own memory for the credentials but, rather than attempting a live sign-in, discovered that the App Store Connect API key was still present at the expected hidden path and used the API key to resume monitoring without ever completing an interactive sign-in. On methodological scope: the qualitative log analysis behind these findings is inherently incomplete, and other notable behaviors may remain undiscovered in the logs.

Table 1: An incomplete survey of open-world evaluations conducted between February 2025 and March 2026.

Evaluation	Length	Human Role	Cost	Agent Capabilities	Agent Limitations
Claude Plays Pokemon [36]	Weeks	Setup-only	Not disclosed	Navigated menus, battled trainers, made story progress	Stuck in Mt. Moon for ∼80 hours; large gap between playing and playing well
AI Village [37]	Months	Setup-only	∼$50K/yr	Built word games; launched a Substack; late-2025 models showed meaningful improvement; sustained multi-week execution	Persistent hallucination and unproductive loops; GUI bottlenecks impeded completion of basic tasks
Project Vend [73, 3]	Weeks	Monitoring	Not disclosed	Phase 2 achieved weekly profit; fixed Phase 1 failure modes	Social-engineering vulnerabilities: agent manipulated into giving away inventory
Cursor Browser [38]	1 week	Setup-only	∼3B tokens ($10K–50K)	Functional Rust rendering engine supporting HTML, CSS, layout, and paint on real sites	Far from production quality; flat swarm collapsed to 2–3 effective agents before a hierarchical reorganization
C Compiler [2]	2 weeks	Monitoring	∼$20K	99% GCC torture-test pass rate; compiled the Linux kernel, PostgreSQL, Redis, FFmpeg, and Doom	Less efficient than GCC with optimizations disabled; ceiling around ∼100K lines of code at which bug fixes broke existing functionality
Epoch knowledge-work tasks [42]	Hours	Setup-only	$20–200/mo	Partial success on Substack porting and web-interface replication	Failed at basic GUI operations; hallucinated data; visual computer-use horizons 40–100× shorter than text-based
Codex Design Tool [79]	∼25 hrs	Setup-only	∼$200	Long-horizon coherence via milestone-based planning and verification	Reported only by authors; no independent evaluation or live demonstration
vinext [39]	∼1 week	Collaborative	∼$1,100	94% Next.js API coverage; 4.4× faster builds; 57% smaller bundles	Target was well-specified with existing test suites; Vite and its RSC plugin provided much of the functionality
Training a Computer [41]	Hours–days	Mixed	$20–200/mo	Human-guided Claude Code generalized to multi-step computations absent from training	Both agents reward-hacked in the fully autonomous condition
Nanochat Autoresearch [40]	Days	Collaborative	<$100	∼100 experiments overnight; reported 11–19% improvements on "Time to GPT-2"	Absolute improvements are small; the agent lacks a mechanism for reasoning about why a change succeeded

C Survey of open-world evaluations: details

This appendix expands on Section 2.3 by describing ten representative open-world evaluations conducted between February 2025 and March 2026, and provides a side-by-side comparison of their reported capabilities, limitations, and costs in Table 1.

1. **Anthropic, Claude Plays Pokemon** [36] (February 2025). Anthropic ran a Twitch livestream in which Claude 3.7 Sonnet played Pokemon Red. Although not a deployment, the experiment was an early instance of situating an agent in a relatively open environment compared to typical benchmarks. It illustrated both progress in AI computer use under minimal scaffolding and the limitations of early-2025 agents: the agent remained stuck on a single level for approximately 80 hours [36].[5]

[5] For context, most children are able to beat the entire game in around 25 hours [80].

2. **AI Digest, AI Village** [37] (April 2025–present). Multiple agents are given individual computer environments and a shared group chat, and tasked with open-ended real-world goals such as charity fundraising, organizing in-person events, and building a Substack presence. The project has documented persistent failure modes such as hallucination, miscalibration, and unproductive loops, alongside notable improvements from late-2025 agents [69].
3. **Anthropic/Andon Labs, Project Vend** [73] (June 2025–present). Anthropic and Andon Labs deployed a Claude 3.7 Sonnet-based agent ("Claudius") to operate a small automated store in Anthropic's office, managing inventory, pricing, and customer interactions over several weeks, and surfacing failure modes around manipulation, prioritization, and real-world decision-making. A second phase [3] expanded the experiment to multiple locations with newer models and included a red-teaming exercise by Wall Street Journal staff. The original phase incurred substantial losses from poor planning, hallucination, and excessive discounting; the follow-up was more profitable, though WSJ staff were still able to jailbreak the agent into giving away inventory [81]. Andon Labs has since initiated a third phase in which a Claude-based agent ("Luna") has been given a three-year lease on a brick-and-mortar store and tasked with operating it profitably, including hiring employees and designing the brand [74].
4. **Cursor, browser experiment** [38] (January 2026). Wilson Lin at Cursor coordinated hundreds of GPT-5.2 agents to build a web browser from scratch over one week. The resulting browser ("FastRender") comprised over a million lines of Rust, including a from-scratch rendering engine. It could render simple websites but was far from production-ready. The experiment is notable for its exploration of hierarchical multi-agent coordination at scale and for characterizing the failure modes that emerge over multi-day agent runs.
5. **Carlini, C compiler** [2] (February 2026). Nicholas Carlini used Claude to build a C compiler from scratch at a cost of approximately $20,000 in API usage. The agent produced a working compiler capable of compiling the Linux kernel and passed a large fraction of standard test suites. The experiment surfaced both strengths (systematic code generation, test-driven iteration) and weaknesses (complex optimization passes, debugging subtle specification violations).
6. **Ho, knowledge-work tasks at Epoch** [42] (February 2026). Anson Ho had Claude Code and ChatGPT Atlas attempt three knowledge-work tasks at Epoch: replicating an interactive web interface for a 40-parameter economic model, writing a 2025 AI-progress article in Epoch's style, and porting an article from Google Docs to Substack and Epoch's website. Formatting bottlenecks and hallucinations emerged as persistent limitations.
7. **Choi, GPT-5.3 Codex design tool** [79] (February 2026). Derrickk Choi of OpenAI ran GPT-5.3 Codex autonomously for 25 hours to generate 35,000 lines of code for a "design tool." The associated report describes planning, memory, and verification behavior but does not substantively analyze the capabilities and limitations of the finished product.
8. **Faulkner, Next.js reimplementation** [39] (February 2026). A Cloudflare engineer used Claude with OpenCode to release *vinext*, a reimplementation of the Next.js frontend framework atop Vite rather than React. The associated writeup reported 94% coverage of Next.js for approximately $1,100 in API costs, but subsequent community analysis [82] identified security limitations and limited generalizability, noting that much of the heavy lifting was provided by the existing testing infrastructure of Vite and Next.js.
9. **Papailiopoulos et al., training a computer** [41] (March 2026). The authors tested whether Claude Code and OpenAI Codex could train a transformer to function as a general-purpose computer. In the fully autonomous condition, both agents failed and produced reward-hacked solutions; in a human-guided condition, Claude Code succeeded and displayed meaningful generalization, including to multi-step computations absent from its training.
10. **Karpathy, Nanochat autoresearch** [40] (March 2026). Karpathy built a simple automation pipeline atop the open-source nanochat project (for GPT-2-level LLM training), giving an agent full autonomy to adjust architecture, hyperparameters, optimizers, and batch sizes in 5-minute increments. In a follow-up, Karpathy reported that the agent reduced the "Time to GPT-2" metric (measured on $8\times$H100 GPUs) by 11% over two days.