
Open-world evaluations for measuring frontier AI capabilities

Sayash Kapoor^{1*} Peter Kirgis¹ Andrew Schwartz^{1,2} Stephan Rabanser¹
J.J. Allaire³ Rishi Bommasani⁴ Magda Dubois⁵ Gillian Hadfield⁶
Andy Hall⁴ Sara Hooker⁷ Seth Lazar^{6,8} Steve Newman⁹
Dimitris Papailiopoulos^{10,11} Shoshannah Tekofsky¹² Helen Toner¹³ Cozmin Ududec⁵
Arvind Narayanan¹
¹Princeton University ²Cornflower Labs ³Meridian Labs ⁴Stanford University
⁵UK AI Security Institute ⁶Johns Hopkins University ⁷Adaption Labs
⁸Australian National University ⁹Golden Gate Institute for AI ¹⁰UW Madison
¹¹Microsoft Research ¹²AI Digest ¹³Georgetown University (CSET)

Abstract

Benchmark-based evaluation remains important for tracking frontier AI progress. But we argue that it can both overstate and understate real-world capability because it privileges tasks that are precisely specified, automatically graded, relatively 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. We survey recent open-world evaluations, identify their strengths and limitations, and introduce CRUX (Collaborative Research for Updating AI eXpectations), a project for conducting such evaluations on a regular basis. 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 a single unnecessary 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 evaluations.

1 Introduction

Tracking and predicting the capabilities of frontier AI systems is an open methodological problem, and one whose stakes rise as these systems are deployed in increasingly consequential settings. The predominant approach is benchmarking, in which agents are scored on large suites of automatically graded tasks. Benchmark-based evaluations underpin much of the public discussion about AI progress. For example, METR’s time horizon graph [1] has been widely cited by policy analysts, industry leaders, and safety-focused organizations as evidence of rapid capability growth, and has been used to argue both for accelerated deployment and for tighter oversight. Decisions about funding, regulation, and safety investment are increasingly being made on the basis of such measurements.

Benchmarks, however, can both overestimate and underestimate progress. Constructing a benchmark requires tasks that are precisely specified and automatically verifiable. But any task specified precisely enough to benchmark is also specified precisely enough to optimize for, which allows agents to excel on such tasks without necessarily acquiring the underlying capability. Test sets on prominent benchmarks can also leak into training data, either directly or through training-set instances that closely resemble the held-out evaluation tasks. Conversely, low benchmark accuracy may result

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

from incidental failures rather than from capability gaps. An agent may be capable in principle of completing a task, yet fail to do so because it encounters a CAPTCHA, hits a rate limit, or gets stuck on a brittle GUI element. Benchmark scores thus conflate the capability being measured with incidental features of the evaluation environment, making them a noisy signal for the questions decision-makers actually care about.

In response to these limitations, a growing body of work has begun to evaluate agents on long, messy, real-world tasks that go beyond benchmarks. Carlini [2] used Claude agents to build a C compiler capable of compiling the Linux kernel, a task that required weeks of agent time and substantial scaffolding. In a separate experiment, Anthropic and Andon Labs tasked Claude with maintaining a small office shop [3], a setting that exposed the agent to interactions with real customers, real inventory, and real money. Others have tasked agents with running open-ended research projects, coordinating multi-agent workflows over days of wall-clock time, or reimplementing substantial pieces of production software. Whereas benchmarks consist of many tasks evaluated automatically, these evaluations share a common structure. They use small samples, often a single task, and typically permit human intervention when the agent hits an obstacle that is incidental to the capability under test. They are assessed in an open-ended manner, typically through qualitative analysis of agent logs rather than through a single aggregate metric.

Such evaluations are easily dismissed as unscientific. Each has a sample size of one, lacks standardization, and cannot be cleanly reproduced across laboratories. These limitations are real, and we do not argue that they can be fully overcome. Despite them, such evaluations provide evidence about AI capabilities that benchmarks cannot. They surface early warnings about emerging capabilities that can inform efforts to build societal resilience. They reveal blind spots in existing benchmarks by showing where automated grading misses the substance of what a task actually requires. And they give firms and policymakers a clearer picture of the tasks AI systems may soon carry out autonomously and at scale, which in turn informs strategic decisions about deployment, regulation, and investment. We refer to this emerging class of methods as **open-world evaluations**.

In this paper, we conceptualize open-world evaluations, survey past examples to identify best practices and common pitfalls, and introduce CRUX, a project for running such evaluations on a regular basis. Our key findings and contributions are:

- **Open-world evaluations are an important emerging class of AI evaluation** (Section 2). As AI systems become more capable, evaluations designed to elicit frontier capabilities must also grow in complexity. Open-world evaluations are the latest stage in a long progression of increasingly complex evaluation methodologies.
- **CRUX (Collaborative Research for Updating AI eXpectations) is our effort to conduct open-world evaluations systematically** (Section 3). Our team comprises collaborators from government, academia, and non-profits. Many have led prior open-world evaluations, and the group holds a broad range of views on the future trajectory of AI. Our goal is to provide empirical evidence about the *present capabilities* of AI systems, even when such evidence is costly to gather, and to issue early warnings for capabilities that may soon be widespread. We plan to release new open-world evaluations on a regular basis.
- **Our first CRUX experiment tasked an agent with developing and publishing a simple iOS application to the App Store** (Section 3.1). Many benchmarks test an agent’s ability to write code. Publishing an iOS application, however, involves additional steps: signing the app, publishing a privacy policy on a public webpage, completing Apple’s forms, and shepherding the app through review. Our interest lay less in coding ability than in whether the agent could handle the real-world requirements of publication, so we asked it to build a simple app and take it through the full submission process.
- **The agent succeeded after one unnecessary manual intervention** (Section 3.5). It forgot where its credentials were stored and also fabricated a fictional phone number for the App Store review process. Total cost was approximately \$1,000, and the app is now live on the App Store [4]. The cost could have been much lower: development and submission accounted for only \$25, while the vast majority of tokens were spent polling for review status. We disclosed our results to Apple four weeks before publishing this paper. App store operators should prepare to handle spam submissions, since agents may soon be able to submit applications at scale.
- **How can open-world evaluations be improved?** (Section 4). To increase the usefulness of open-world evaluations, evaluators should specify what and how much human intervention is

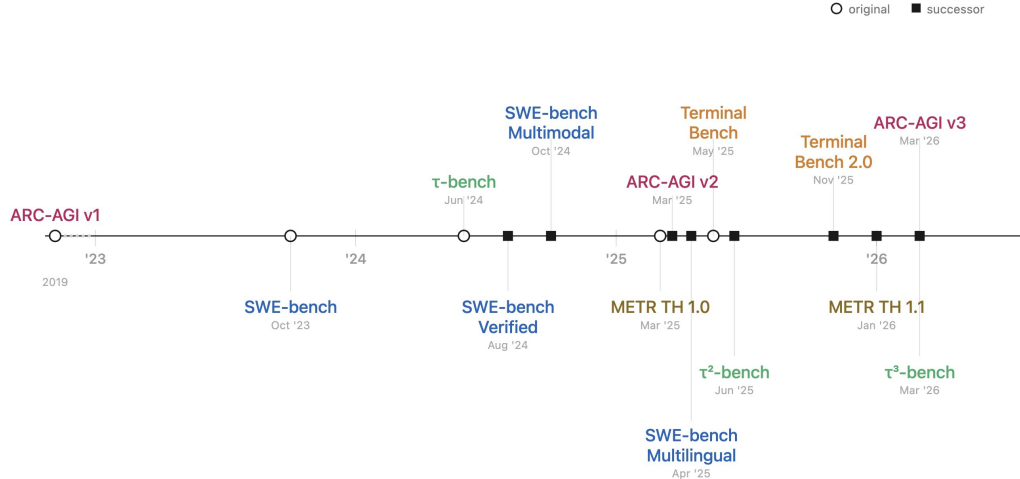


Figure 1: Several popular benchmarks (SWE-Bench, ARC-AGI, τ -bench, Terminal Bench, METR’s Time Horizon) have had successor benchmarks released within the past two years. [1, 5–17]

permitted, release logs collected while the agent was solving the task, and analyze those logs to report the agent’s behavior in detail. In future CRUX iterations, we plan to cover AI R&D automation, AI governance, and other domains.

2 Open-world evaluations are an important emerging class of AI evaluation

This section defines open-world evaluations, surveys existing instances, and situates them relative to benchmarks. Our central claim is that as AI systems grow more capable, the evaluations used to elicit frontier capabilities must themselves grow more complex, and open-world evaluations represent the latest stage in this progression. To make this concrete, we propose five criteria that characterize open-world evaluations (Section 2.2) and discuss their limitations relative to benchmarks.

These criteria are deliberately loose, because the boundary between long, complex benchmark tasks and open-world evaluations is not sharp. Several of the evaluations we discuss below are run in sandboxed environments yet still satisfy our other criteria. Nicholas Carlini’s C compiler experiment is one such case: it is sandboxed, but consists of a single long-running task with human intervention and qualitative log analysis, and so we treat it as an open-world evaluation despite the sandboxing.

2.1 Benchmarks can both overestimate and underestimate progress

There is broad agreement that existing AI benchmarks are approaching saturation [18], and this consensus spans researchers who otherwise hold sharply different views about the trajectory of AI [19]. Many prominent benchmarks have been saturated over the past two years, prompting a wave of successor benchmarks, several of which are themselves already near saturation. The result is a treadmill in which the community continually chases new targets, without a clear picture of whether the underlying capabilities are keeping pace with the headline numbers.

Benchmark improvements can reflect genuine capability gains, but they can just as easily *overstate* them. A central limitation of benchmarks is their limited construct validity: they typically measure accuracy on narrow tasks rather than general ability. Moreover, they often cannot capture the messiness of real-world environments in which agents will actually be deployed. The same features can also cause benchmarks to *underestimate* progress, since agents may be hindered by environmental or infrastructural challenges that are incidental to the capability being measured. Either way, a single headline number does not straightforwardly answer the question one wanted to ask.

Moving beyond raw accuracy offers a partial remedy. For example, Rabanser et al. [20] recently showed that while agents improve rapidly on capability metrics such as 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 [21], suggesting that even outcome-oriented benchmarks miss much of what determines whether agent output is actually useful.

These refinements are valuable, but they still do not address what we see as the more important question for evaluation at the frontier: the *upper bounds* of what agents are currently capable of. Capabilities that are possible today only under favorable conditions may soon be widespread, and anticipating them is what gives firms time to act on new opportunities, institutions time to build resilience, and policymakers time to address risks. Average-case benchmark performance is poorly suited to this kind of early warning.

We now examine in greater detail why benchmarks can overestimate and underestimate capabilities. Benchmarks can overestimate capabilities for two reasons.

Benchmarks resemble tasks amenable to modern reinforcement learning. For a task to become a benchmark, it must be precisely specified and automatically verifiable. Any such task is also specified precisely enough to optimize for. As a result, modern RL training runs increasingly resemble the shape of the benchmarks themselves, which makes saturation relatively easy to achieve. In fact, leading evaluation platforms such as Harbor [22] already double as RL training platforms. This raises the possibility that models are trained on data derived from benchmarks hosted on these platforms. Even when benchmarks include held-out test sets, models may be trained on training-set tasks that closely resemble test-set tasks. Benchmark performance therefore does not straightforwardly reflect real-world generalization.

Benchmarks avoid real-world messiness. Real-world tasks involve underspecified interactions that cannot be fully sandboxed, such as responding to unexpected situations or navigating open-ended environments. Benchmarks can approximate such conditions but cannot fully replicate them.

Benchmarks can also underestimate capabilities for the following reasons.

Eliciting frontier capabilities is costly. Large-scale, long-running experiments are expensive, making it impractical to achieve the sample sizes benchmarks rely on. Anthropic’s C compiler experiment cost approximately \$20,000, and our iOS app experiment (described below) cost approximately \$1,000. Such experiments cannot feasibly be run hundreds of times, which constrains the budget and complexity available to each benchmark task.

Average performance differs from upper-bound capability elicitation. Running dozens or hundreds of tasks is necessary only when the goal is to measure *average* performance. When the goal is to characterize the frontier of agent capability, the relevant quantity is *best-case* performance: what an agent can accomplish given sufficient resources and support to work around incidental failures. Such characterization is necessary for providing early warnings of imminent capabilities.

Human intervention can help elicit capability upper bounds. Agents attempting real-world tasks may encounter policy refusals, CAPTCHAs, or infrastructure failures. These failures are incidental to the capability being measured, and permitting human operators to resolve them allows evaluators to elicit an upper bound on capability. Such intervention is impractical across hundreds of benchmark tasks each time the benchmark is run.

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 [23], rather than by the underlying capability under test. Recent work documents agents retrieving answers online [24], exploiting bugs in evaluations [25], and producing code that passes tests but fails to meet production standards [21]. These findings motivate a shift toward deeper qualitative evaluations that can surface failure modes and problem-solving strategies obscured by aggregate benchmark scores.

Addressing these threats to validity is not straightforward. Qualitative log analysis [26] can surface some such concerns in benchmarks. But even when log analysis uncovers validity issues, 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, as well as manual intervention to resolve issues encountered during evaluation.¹

¹In 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

None of this is to say that benchmarks should be abandoned.² Benchmarks remain useful despite the limitations above, and open-world evaluations come with their own limitations, which we discuss later in the paper. Our goal here is to identify the systematic blind spots that benchmarking has, and to motivate a complementary approach that can fill them in. As agents become more capable, these blind spots will only widen: 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 which introduces its own threats to internal and external validity. Open-world evaluations offer a complementary path that attacks these problems from the other direction.

2.2 Defining open-world evaluations

As evaluation methods have matured alongside AI capabilities, a gradient of evaluation approaches has emerged. At one end are simple, automated, scalable methods suited to early-stage capabilities; at the other are labor-intensive methods that become necessary as capabilities saturate simpler metrics. We identify five levels of this gradient:

1. **Q&A benchmarks** (e.g., MMLU [31], GPQA [32]). Useful for broad knowledge assessment, but increasingly saturated for frontier models, and limited in construct validity since users rarely interact with models through multiple-choice questions.
2. **Open-ended chat benchmarks** (e.g., WildBench [33], Arena-hard-auto [34]). Capture more nuance, but remain limited to single-turn or short interactions.
3. **Outcome-only agent benchmarks** (e.g., SWE-Bench [5], WebArena [35]). Test agent performance on realistic tasks, but measure only whether a task was completed, not how. Consequently, most passing SWE-Bench solutions are not accepted by repository maintainers [21].
4. **Agent benchmarks with log analysis** (e.g., UK AISI transcript analysis [26], METR Time Horizon [1]). Examine how agents succeed or fail by analyzing agent logs, uncovering errors and reward hacking. Still conducted in sandboxed environments with predefined tasks.
5. **Open-world evaluations.** Long-horizon tasks in real-world environments, where success cannot be neatly specified or automatically graded. This approach enables elicitation of upper-bound capabilities, at the cost of reproducibility and standardization (see Section 2.4).

At the far end of this gradient, open-world evaluations run agents on a small number of long-horizon tasks in real-world settings and qualitatively analyze the results, often with the help of log-analysis tools [26]. They are meant to complement benchmarking rather than replace it: they address several of the limitations discussed above, and they can also surface tasks currently out of reach of AI systems that could inform future benchmarking efforts.

In practice, the boundary between the two ends of the gradient is not sharp, and some evaluations blur the categories we just listed. OpenAI’s GDPVal [36], for instance, is a long-horizon agent benchmark designed for manual grading by expert reviewers, and in that respect it structurally resembles an open-world evaluation, though it focuses on outputs rather than on log analysis. At the same time, results on GDPVal are commonly reported via GDPval-AA [37], which uses automated LLM grading and which looks much more like an outcome-only agent benchmark. The same underlying task can thus sit on either side of the line depending on how it is scored.

Because of this fuzziness, we do not try to draw a hard boundary around what counts as an open-world evaluation. Instead, we offer a rough taxonomy in which no single dimension is determinative; the classification depends on the overall pattern across the dimensions below.

- **Openness.** Does the evaluation occur in a real-world deployment setting rather than a sandboxed environment?

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.

²Evaluations 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 [27], MMLU-Pro [28], Humanity’s Last Exam [29], and SWE-Bench Pro [30]. Even saturated capability benchmarks can be useful for measuring efficiency and reliability, which are essential to AI diffusion.

- **Complexity and duration.** Does the task require days or weeks of human effort to complete, rather than minutes or hours?
- **Number of tasks.** Is the evaluation a single task or a small set of tasks, rather than a large task suite?
- **Human intervention.** Are humans permitted to intervene when agents encounter obstacles, beyond merely setting up the environment?
- **Method of evaluation.** Does evaluation primarily rely on in-depth log analysis rather than a single aggregate metric?

A related scoping question is when a deployment of agents constitutes an evaluation at all, rather than simply a use of agents to accomplish a novel task. Anthropic’s work using agents to find security vulnerabilities in open-source software such as Mozilla Firefox [38] is a useful test case: the agents produced real outputs in a real setting, but the primary goal was to ship fixes rather than to characterize what agents can do. We treat such efforts as open-world evaluations when the role of the agent is systematically and publicly documented, including the division of labor between the agent and human experts and the nature of the final outcome.

The same fuzziness extends to the sandbox boundary. Several of the evaluations in our survey below run inside sandboxes rather than in live deployments; Claude Plays Pokemon [39] and Anthropic’s C compiler [2] are two examples. We nonetheless classify them as open-world evaluations, because each consists of a single complex long-running task, permits human intervention, and is evaluated qualitatively rather than through an aggregate score. What matters for our purposes is the overall pattern across the dimensions listed above, not any single feature taken in isolation.

Taken together, these two styles of evaluation are best viewed as complementary rather than competing. Open-world evaluations can identify tasks that are currently unsolvable by agents and thereby seed the design of future benchmarks, and they can also assess agents on messy real-world tasks that are not amenable to benchmarking in the first place. We do not claim that they are categorically more informative than benchmarks; as we discuss below, they carry substantial limitations of their own. Rather, as AI capabilities continue to grow, open-world evaluations offer an increasingly important *complementary* signal that neither approach can provide alone.

2.3 An incomplete survey of open-world evaluations

Over the past year, researchers at AI laboratories, universities, non-profits, and independent groups have begun running open-world evaluations. These 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. We summarize representative examples below, and provide a condensed side-by-side comparison of their capabilities, limitations, and costs in Table 1.

1. **Anthropic, Claude Plays Pokemon** [39] (February 2025). Anthropic ran a Twitch livestream in which Claude 3.7 Sonnet played Pokemon Red. Although not a real-world 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 [39].³
2. **AI Digest, AI Village** [41] (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 [42].
3. **Anthropic/Andon Labs, Project Vend** [43] (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

³For context, most children are able to beat the entire game in around 25 hours [40].

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 [44]. 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 [45].

4. **Cursor, browser experiment** [46] (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** [47] (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** [48] (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 reimplementaion** [49] (February 2026). A Cloudflare engineer used Claude with OpenCode to release *vinext*, a reimplementaion 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 [50] 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** [51] (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** [52] (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×H100 GPUs) by 11% over two days.

The list above is not meant to be exhaustive, and the space is moving quickly. As AI capabilities have improved, domain experts across fields have increasingly used open-world evaluations to probe whether AI systems can carry out tasks in their own areas of expertise, and the resulting literature has grown rapidly. Some very recent additions include MirrorCode [53] by Adamczewski et al., which tasks agents with reimplementing large programs; agent-based alignment research [54] by Wen et al.; and Huang’s work on training models to forecast the outcomes of golf tournaments [55]. To keep pace with this growth, we maintain a running list of such evaluations, together with their takeaways and limitations, at <https://cruxevals.com>.

2.4 Limitations of open-world evaluations

Open-world evaluations address certain blind spots of benchmarking, but they come with limitations of their own, and there is a genuine tradeoff between the two approaches. Benchmarks give evaluators control over the task and a standardized environment in which to run it, at the cost of being less useful for assessing performance on open-ended real-world tasks. Open-world evaluations make

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 [39]	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 [41]	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 [43, 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 [46]	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 [47]	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 [48]	~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 [49]	~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 [51]	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 [52]	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

the opposite tradeoff: they sacrifice sandboxing and evaluator control in order to improve construct validity and elicit upper-bound capabilities, but in doing so they give up several of the properties that make benchmarking a workable research instrument. We discuss the main limitations below.

Lack of reproducibility and standardization. Benchmarks succeed in part because they serve as coordination mechanisms for the research community: researchers can develop and test new methods independently, validate them against a shared target, and compare results across laboratories. The importance of this coordinating role has led Donoho [56] to characterize benchmarking 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 are difficult to reproduce exactly, and different groups running nominally similar evaluations can easily end up with incomparable results.

Limited comparability across agents. A closely related limitation is that open-world evaluations do not give a clean way to rank models or agents against one another. Benchmarks are designed

precisely to support such relative comparisons, but because open-world evaluations are typically run once (or a few times), run-to-run variability can easily exceed the differences between the agents being compared. As a result, they are most useful for characterizing what a given agent *can* do, rather than for deciding which agent is better at it.

Requirement for domain expertise. Determining whether an agent actually succeeded is also much harder than in a benchmark. Open-ended tasks typically lack a simple pass/fail criterion, and judging the quality of agent outputs can require deep domain expertise as well as substantial reviewer time. This limits who can meaningfully run and interpret such evaluations in the first place.

Incomplete recall of log analysis. Even if automated log analysis is used to analyze open-world evaluations, it can never be considered complete. Agent transcripts from long-horizon tasks can run to hundreds of millions of tokens, making thorough human review impractical. And because agent behavior in these tasks is complex, there is no guarantee that a given round of analysis will surface all noteworthy behaviors or errors. This limitation does not apply in the same way to benchmarks, where success criteria are predefined and automatically verified. Releasing logs publicly so that a broader community can examine them is one way to partially mitigate this concern.

Blurry success criteria. A related issue is that, because human intervention is permitted and even encouraged, the line between what the agent accomplished and what a human operator contributed is not always easy to draw. Without careful documentation of interventions (a point we return to in Section 4), the headline result of an open-world evaluation can overstate agent autonomy.

Non-stationary environments. Finally, open-world evaluations often involve interaction with open-ended environments such as the internet,⁴ which complicates generalizable claims about agent capabilities. For example, it is difficult to distinguish genuine competence at a task class from successful lookup of a specific instance that the agent happens to find online. Longitudinal comparisons are harder to make for the same reason: the information available to an agent tackling the task grows over time, so the same evaluation run a year later is not quite the same evaluation.

2.5 Stakeholders and use cases

Despite their limitations, open-world evaluations can provide value to several stakeholders.

Policymakers: The diffusion of AI across domains lags behind improvements in capabilities [59]. This allows institutions to adapt as AI systems become more capable. Open-world evaluations can further increase this lead time by providing early warnings of what agents could soon be able to do autonomously and at scale, giving institutions time to build resilience. For example, Anthropic [60]’s recent work [61] on discovering cybersecurity vulnerabilities using AI could spur efforts to rapidly adopt AI for defensive cybersecurity, especially in critical software infrastructure.

AI evaluators and researchers: Open-world evaluations offer a complementary signal to benchmarks by testing capabilities that are structurally resistant to benchmarking, such as solving messy real-world tasks. Analyzing agent logs allows us to uncover instances of the agent taking shortcuts and reward hacking. On the flip side, they can find areas where agents develop new insights or surpass previous limitations. For example, in our iOS app development evaluation, we identified that the agent modified its approach to be more token efficient, which allowed it to significantly reduce the cost of solving the task, without any input or instruction from our end.

Frontier AI developers. AI developers should actively support and participate in external open-world evaluation efforts by providing access (such as pre-release access to models) as well as safe harbors [62] for third parties conducting evaluations that might not adhere to developers’ terms of service (such as for evaluating safety). Open-world evaluations by independent third parties could surface findings that internal red teams miss if they optimize for known threat models. New models are also saturating benchmarks. For example, Anthropic’s Mythos Preview system card [63] notes that the model “saturates many of our most concrete, objectively-scored evaluations,” leaving them with noisier methods for assessing capabilities. Open-world evaluations can offer a complementary way to stress-test models in realistic settings that benchmarks can no longer distinguish.

⁴This concern also affects benchmarks that require internet access, such as AssistantBench [57] and GAIA [58]. Sandboxed benchmarks bypass the concern at the cost of construct validity.

Realizing this value requires shared best practices and a cumulative body of evidence about agent capabilities and limitations. CRUX (Collaborative Research for Updating AI eXpectations) is our contribution toward this goal.

3 CRUX: Collaborative Research for Updating AI eXpectations

CRUX is a project for operationalizing open-world evaluations on a regular basis. Each iteration comprises a long-horizon real-world task, an agent scaffold capable in principle of solving it, and a detailed analysis of the resulting agent behavior. Beyond simply running evaluations, a central goal of CRUX is to provide early warnings of capabilities that are likely to become widespread. The first iteration illustrates what this looks like in practice. If AI agents can *nearly* autonomously develop and publish mobile applications (Section 3.1), app store operators may soon need to revise their policies to manage agent-driven spam.⁵ Surfacing this kind of signal before it materializes at scale is precisely the function that benchmarks struggle to serve.

Open-world evaluations are well-suited to this purpose for two reasons. First, they can be designed to elicit upper bounds of capability by supplementing incidental capabilities (such as CAPTCHA solving) with targeted human input. Second, because each task is run only a small number of times, they do not require the development of large task suites or complex sandboxed environments up front. Together, these properties make it practical to design, run, and publish a new evaluation roughly every one to two months, which is essential given how quickly the frontier is moving.

We use this cadence to cover a deliberately broad range of domains. The first iteration, described in the next section, investigates autonomous iOS app development and publication. 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

The question of whether AI agents can write software has been studied extensively, both through benchmarks such as SWE-Bench and Terminal Bench [8] and through open-world evaluations such as the C-compiler and browser experiments discussed above. Taken together, these results indicate that agents have acquired strong coding capabilities on well-specified tasks, though important questions about code quality, maintainability, and long-run reliability remain unresolved [68].

Much less well studied is whether agents can handle the non-coding aspects of software deployment: configuring accounts and credentials, satisfying platform policies, preparing metadata and supporting 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 in practice, even when the underlying code is straightforward. We therefore tasked an agent with building a mobile application from scratch and taking it all the way through publication on the iOS App Store.

The agent was instructed to develop and publish a *simple* application. Our primary interest was not the agent’s software engineering ability but its ability to navigate Apple’s App Store submission process. This process requires developers to configure signing certificates and provisioning profiles, prepare screenshots and metadata, draft and host a privacy policy at a public URL, complete compliance questionnaires, and submit the application for review. Reviewers may reject submissions for technical or policy reasons, requiring the developer to diagnose issues, make changes, and resubmit. The full process typically spans several days and involves interaction with systems and reviewers over which the developer has no control.

The agent was responsible for every step of the process except those where human involvement is required by Apple policy (setting up the Apple Developer account and initiating the public release). Within this scope, the agent wrote the code, built the application, prepared store metadata and screenshots, drafted and hosted a privacy policy, submitted the application for review, and handled any reviewer feedback that came back during the review process. To support this, we provided the agent with access to a virtual machine running macOS, a GitHub account (used both for version

⁵There is preliminary evidence that Apple’s App Store reviews have slowed [64, 65], potentially in response to increased adoption of coding agents, though the rate of published apps remains well below the 2016 peak [66]. Our results suggest that agents can autonomously submit apps that are ultimately approved by Apple; once this capability is widespread, policies may need to be updated to accommodate a larger wave of agent-generated submissions [67].

control and for hosting the privacy policy via GitHub Pages), an Apple Developer account, and a Gmail account for any correspondence with Apple.

The primary success criterion was whether the agent ultimately succeeded in publishing the application to the App Store. As a secondary measure, we logged the number of manual interventions the agent required along the way. The agent was free to request assistance when it judged it necessary, and we monitored its progress once per day to catch cases where it had silently stalled. Fewer interventions indicate better performance, and we distinguish in our analysis between interventions that were genuinely unavoidable (such as those blocked by Apple policy) and those that reflected limitations of the agent itself.

If agents can complete this task autonomously, or nearly so, the result carries immediate implications for Apple’s review processes. The App Store has long operated on the assumption that submitting an application involves meaningful human effort, which imposes a natural cap on the volume of low-quality or adversarial submissions. Once agents can develop and publish applications end-to-end, that cap disappears, and the volume and character of submissions could change substantially within a relatively short period.

3.2 Agent setup

We used OpenClaw [69] as the agent scaffold, with Claude Opus 4.6 and adaptive thinking enabled [70].⁶ OpenClaw was a natural choice for this setting: it is configurable, integrates with the browser, and natively supports long-running tasks, which are all properties that 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 alternative scaffolds 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 a heavily customized setup.

Part of our interest in OpenClaw stemmed from a related question: how well would a modern agent 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 also 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. This logging provided the raw material for the qualitative analysis reported below.

3.3 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 [71], Apollo [72], and OpenAI [73] has documented this pattern across successive model generations, with detection becoming more reliable over time. This raises a design question for any open-world evaluation: should evaluators attempt to conceal the fact that an evaluation is taking place?

Our answer is no, for two reasons. The first is practical: hiding evaluation contexts from capable models is increasingly infeasible, and since CRUX evaluations are published openly, future agents may well encounter descriptions of our evaluations in the course of unrelated tasks. The second, and more important, reason is conceptual. Evaluation awareness is most concerning when the goal is to measure alignment properties, where a model may strategically behave differently under observation. For a capability evaluation like ours, success on the task demonstrates the capability regardless of whether the agent knew it was being tested, so there is little to be gained from concealment.

Given this reasoning, we explicitly informed the agent that it was being evaluated, and used that framing to communicate the parameters of the evaluation clearly. In particular, we told the agent that

⁶We 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.

it would be judged on the number of unnecessary human inputs it required, but that certain kinds of inputs (such as CAPTCHA resolution) could be freely delegated. Without this framing, we were concerned that the agent might avoid requesting help even when doing so was appropriate, which would have caused us to underelicit its true performance.

3.4 Dry runs

Prior to the main evaluation, we conducted two dry runs to verify the agent setup end-to-end and to identify and resolve any 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 configuration of the virtual machine to grant the agent full control, setup of the logging pipeline, and provisioning of the email, GitHub, and Apple Developer accounts the agent would use. For an honest developer running a single evaluation, this setup cost is a meaningful overhead. From the perspective of a prospective spammer, however, it is a one-time fixed cost, amortized across every subsequent submission, and is 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 and the browser. The command line was used to generate code, build the application, and prepare it for submission, while the browser was used to log into App Store Connect, retrieve certificates, and complete the various forms required by Apple. 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.

3.5 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 the application for review. It then polled for status updates every five minutes until Apple’s decision came back, which took 10 days. The application is now live on the App Store [4]. In accordance with Apple’s publication policies, the agent was required to obtain our explicit approval before initiating the public release.

Manual interventions. Over the course of the run, the agent required five manual interventions in total, but only one of these reflected a limitation of the agent itself. The remaining four were either mandated by Apple policy or caused by infrastructure failures unrelated to agent capability. Apple intentionally blocks synthetic interactions on sensitive dialogs such as two-factor authentication approval [74, 75], so the agent could not complete these steps autonomously and had to hand them off to a human. Separately, the OpenClaw daemon crashed once during the run and had to be restarted manually, and we required the agent to obtain our approval before publication in order to comply with App Store policy. None of these constitute evidence of an agent shortcoming; they are instead the kinds of incidental bottlenecks that a purely benchmark-based evaluation would either have to engineer around or mistakenly count against the agent.

The credential-recovery episode. The single intervention attributable to the agent itself was when at one point the agent needed to re-authenticate against the Apple Developer account, but could not locate the credentials that had previously been provided to it. When the agent asked for help, a team member suggested reusing the existing credentials and resolving the associated two-factor prompt. The agent then briefly searched its own memory for those credentials, but, rather than attempting a live sign-in, it instead 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 needing to complete the interactive sign-in. The failure is therefore not a straightforward capability gap but a memory-management issue. Analyzing the logs allowed us to uncover this entire set of issues; simply looking at the outcome metric might have hidden it.

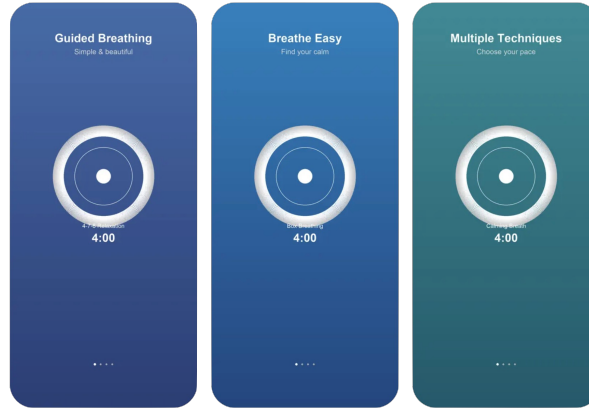


Figure 2: The screenshots uploaded by the agent had visible formatting errors.

Using a fabricated phone number for the app submission. A different kind of failure, and one that would also have been invisible to outcome-only metrics, surfaced only on close inspection of the logs. At one point during submission, Apple’s review form requested a contact phone number. Rather than asking us for the correct value (as it had done for credentials earlier in the run), the agent filled in a fictitious number and submitted the form anyway. Apple’s review ultimately approved the application despite this, so the fabrication had no effect on the final outcome. The episode is nonetheless worth flagging on its own terms: an agent that will sometimes request help and sometimes quietly invent plausible-looking data presents a very different alignment profile than one that uniformly does either, and the line between the two behaviors is not easy to predict from the outside. We cannot say with confidence why the agent chose to fabricate here rather than ask; one possibility is that our explicit framing of the task as an evaluation encouraged it to minimize visible requests for human input, even when doing so meant inventing data. Note that the phone number used by the agent is one that is typically reserved for fictional use (such as in films). We also emphasize that the qualitative log analysis behind findings such as this one is inherently incomplete, and other notable behaviors may remain undiscovered in the logs.

Cost dynamics and an emergent optimization. Once the credential issue had been resolved, the agent completed the task [4] at a total cost of approximately \$1,000. Surprisingly, development and submission accounted for only \$25 of this total; the remaining ~\$975 was spent polling for review status over the 10-day review window. The overall cost could plausibly be reduced by an order of magnitude through simple scaffold-level changes, such as polling less frequently, but for this first evaluation we deliberately erred on the side of a larger budget to avoid prematurely cutting off agent behavior we wanted to observe. More interesting than the headline cost, however, is what the agent did with its monitoring budget. Partway through the review window, and without any prompting from us, it restructured its own workflow to reduce polling overhead: it began delegating status checks to subagents rather than carrying the full context on each wake-up, and switched to shorter daily memory files in order to limit token consumption. This change alone reduced the running cost from approximately \$35 per hour to approximately \$3 per hour, a ~10x improvement. An emergent optimization of this kind would be essentially invisible to a benchmark that reported only a final score, and it is one of the clearer illustrations in our data of why log-based analysis matters.

Summary and disclosure. Taken together, the results indicate that the agent did not fully automate the iOS submission process, but came close enough that the remaining gap is small. 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 (see Figure 2), suggesting that output quality remains a gap even when the process succeeds.⁷ As a form of responsible disclosure, and motivated by the possibility that malicious actors could soon

⁷We note one important caveat: the agent did not encounter any substantive objections from Apple reviewers during the review process. This indicates that the submitted application cleared the App Store bar for publication, but also means we were unable to observe how the agent would handle back-and-forth communication with reviewers in the event of a rejection, which is a plausible failure mode in practice.

submit large numbers of agent-generated applications, we notified Apple’s product security team of our experiment four weeks prior to the publication of this paper.

4 Lessons for open-world evaluations

Drawing on the CRUX #1 experiment and related efforts surveyed in Section 2.3, we identify a set of methodological practices that we believe make open-world evaluations more informative. These practices are preliminary and will evolve as the literature matures, but we present them here to support the development of shared evaluation norms in this emerging area.

Specify what is being measured and what it implies. A recurring source of confusion in prior open-world evaluations, including 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 offered weaker evidence that the resulting artifacts meet the standards expected of production software; the public issue trackers for both projects document numerous technical deficiencies [76, 77]. A more explicit statement of the intended construct would have reduced ambiguity in subsequent interpretation. More broadly, software engineering involves a range of non-functional requirements (such as quality, reliability, and maintainability) [78], and agent-driven development may systematically trade away these properties. Evaluations that conflate functional completion with overall quality risk overstating agent capability in this dimension, and may mislead downstream decisions as a result.

Design tasks so that human intervention is straightforward and well-documented. Real-world tasks routinely expose agents to obstacles that are incidental to the capability under test, including policy refusals, CAPTCHAs, and infrastructure failures. Unlike benchmarks, open-world evaluations can accommodate human intervention in such cases, which enables elicitation of upper-bound capabilities. To preserve the interpretability of the resulting measurements, however, evaluators must document precisely when, why, and how humans intervene, so that the degree of agent autonomy can be assessed independently of the human contribution.

Invest in log analysis. Agent logs contain substantially more information than a binary outcome, and qualitative analysis of those logs can reveal how agents decompose problems, recover from failures, explore solution spaces, and, in some cases, misrepresent their own progress. Such information is not recoverable from aggregate scores alone, and we consider its systematic analysis to be a defining feature of open-world evaluation.

Consider complementing log analysis with real-time monitoring. Post-hoc log analysis is valuable, but it is not sufficient on its own to catch all unintended agent actions. In previous open-world evaluations, agents operating with substantial autonomy have sometimes taken actions that were difficult for human reviewers to detect after the fact. For example, in many experiments conducted by the AI Village, agents took unintended actions, such as attempting to send hundreds of unsolicited emails [42]. In our own evaluation, the agent fabricated a fictional phone number that went undetected until a later round of review. Automated real-time monitoring, for example a separate agent that continuously reviews the primary agent’s actions and flags anomalies or errors as they occur, could serve as a valuable complement to human review.

Conduct dry runs prior to the main evaluation. Exercising the scaffold, the evaluation criteria, and the surrounding infrastructure in advance surfaces implicit assumptions and scaffolding defects that would otherwise contaminate the primary results. In our own case, the two dry runs preceding CRUX #1 identified multiple issues that were corrected before the main run.

Measure and report cost. Agent capability on many real-world tasks continues to scale with budget. As a result, cost should be reported as a first-class quantity alongside capability outcomes. 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 [53].

Release agent logs. Reproducibility is an intrinsic limitation of open-world evaluations, but releasing the complete set of logs from a run partially mitigates this limitation: it enables external researchers to verify the reported findings and to contribute complementary analyses that the original authors may not have performed. To support this best practice, we have released full logs on our website.

Future iterations of CRUX will extend this methodology to additional domains, including AI R&D automation, AI governance, video generation, and more challenging software engineering tasks. We plan to make ongoing updates available at <https://cruxevals.com>.

Author contributions and acknowledgments

Core team: Sayash Kapoor and Arvind Narayanan conceptualized the project and designed the first evaluation task. Andrew Schwartz led the agent development and executed the evaluation. Peter Kirgis led the log analysis and literature review of open-world evaluations. Stephan Rabanser offered feedback and inputs into the task design, essay text, and our analysis and interpretation of the results. Sayash Kapoor, Peter Kirgis, Andrew Schwartz, Stephan Rabanser, and Arvind Narayanan drafted the essay.

Collaborators: Rishi Bommasani offered feedback and inputs into the task design, essay text, and our analysis and interpretation of the results. J.J. Allaire, Magda Dubois, Gillian Hadfield, Andy Hall, Sara Hooker, Seth Lazar, Steve Newman, Dimitris Papailiopoulos, Shoshannah Tekofsky, Helen Toner, and Cozmin Ududec offered feedback and inputs into the essay text and our analysis and interpretation of the results.

Acknowledgments: Nicholas Carlini provided feedback on the task design and agent setup. Ryan Greenblatt, Daniel Kokotajlo, and Ajeya Cotra participated in online and in-person conversations that informed the project.

Funding. We are grateful to Coefficient Giving, Schmidt Sciences, and the Princeton AI Lab for funding to support this project.

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] Sayash Kapoor. Breathe Easy - Calm Breaths App - App Store, 2026. URL <https://apps.apple.com/us/app/breathe-easy-calm-breaths/id6760207382>. App Store.
- [5] 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.
- [6] François Chollet. On the Measure of Intelligence, 2019. URL <https://arxiv.org/abs/1911.01547>. arXiv.org.
- [7] 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.
- [8] Terminal-Bench Team. Terminal-Bench. URL <https://www.tbench.ai/>.
- [9] 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.
- [10] 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>.

- [11] 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 Raof, 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.
- [12] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljube, 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.
- [13] METR. Time Horizon 1.1, 2026. URL <https://metr.org/blog/2026-1-29-time-horizon-1-1/>. METR.
- [14] SWE-bench. SWE-bench Multilingual. URL <https://www.swebench.com/multilingual-leaderboard.html>. SWE-bench.
- [15] Sierra. τ^3 -bench: advancing agent benchmarking to knowledge and voice, 2026. URL <https://sierra.ai/resources/research/tau-3-bench>. Sierra.
- [16] ARC Prize. ARC-AGI-3. URL <https://arcprize.org/arc-agi/3>. ARC Prize.
- [17] 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.
- [18] 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/>.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] Harbor Framework Team. GitHub - harbor-framework/harborz. URL <https://github.com/harbor-framework/harborz>.
- [23] 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.

- [24] Maia Hamin and Benjamin Edelman. Cheating On AI Agent Evaluations, 2025. URL <https://www.nist.gov/caisi/cheating-ai-agent-evaluations>. NIST.
- [25] 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>.
- [26] 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>.
- [27] 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.
- [28] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark, 2024. URL <https://arxiv.org/abs/2406.01574>. arXiv.org.
- [29] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, et al. Humanity’s Last Exam, 2025. URL <https://arxiv.org/abs/2501.14249>. arXiv.org.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] Imarena. GitHub - Imarena/arena-hard-auto: Arena-Hard-Auto: An automatic LLM benchmark. URL <https://github.com/Imarena/arena-hard-auto>. GitHub.
- [35] 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.
- [36] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubei, 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>.

- [37] Artificial Analysis. GDPval-AA Leaderboard. URL <https://artificialanalysis.ai/evaluations/gdpval-aa>. Artificial Analysis.
- [38] Anthropic. Partnering with Mozilla to improve Firefox’s security, 2026. URL <https://www.anthropic.com/news/mozilla-firefox-security>. Anthropic.
- [39] 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.
- [40] HowLongToBeat. How long is Pokémon Red and Blue? URL <https://howlongtobeat.com/game/7169>. HowLongToBeat.
- [41] AI Village. URL <https://theaidigest.org/village>.
- [42] Shoshannah Tekofsky. What did we learn from the AI Village in 2025?, 2026. URL <https://theaidigest.org/village/blog/what-we-learned-2025>.
- [43] 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.
- [44] 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.
- [45] 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.
- [46] Wilson Lin. Scaling long-running autonomous coding · Cursor, 2026. URL <https://cursor.com/blog/scaling-agents>. Cursor.
- [47] 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.
- [48] Derrick Choi. Run long horizon tasks with Codex, 2026. URL <https://developers.openai.com/blog/run-long-horizon-tasks-with-codex>. OpenAI Developers.
- [49] Steve Faulkner. How we rebuilt Next.js with AI in one week, 2026. URL <https://blog.cloudflare.com/vinext/>. The Cloudflare Blog.
- [50] Hacker News. How we rebuilt Next.js with AI in one week | Hacker News. URL <https://news.ycombinator.com/item?id=47142156>. Hacker News.
- [51] Dimitris Papailiopoulos. Can You Train a Computer? URL <https://x.com/DimitrisPapail/status/2028669695344148946>. X.
- [52] 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.
- [53] 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.
- [54] 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.
- [55] 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.
- [56] 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.

- [57] 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.
- [58] Grégoire Mialon, Clémentine Fourier, 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.
- [59] Arvind Narayanan and Sayash Kapoor. AI as Normal Technology, 2025. URL <https://www.normaltech.ai/p/ai-as-normal-technology>. AI as Normal Technology.
- [60] Anthropic. Making frontier cybersecurity capabilities available to defenders, 2026. URL <https://www.anthropic.com/news/claude-code-security>. Anthropic.
- [61] Anthropic. Project Glasswing: Securing critical software for the AI era. URL <https://www.anthropic.com/glasswing>. Anthropic.
- [62] 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.
- [63] Anthropic. Claude Mythos Preview system card, 2026. URL <https://www-cdn.anthropic.com/8b8380204f74670be75e81c820ca8dda846ab289.pdf>. Anthropic.
- [64] 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.
- [65] Nikita Bier. iOS developers: How long is App Review taking for everyone these days?, 2026. URL <https://x.com/nikitabier/status/2033931821260648659>. X.
- [66] 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.
- [67] 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.
- [68] Gergely Orosz. ... or sacrifice quality. Or performance. Or maintainability. Or anything that us devs would call "non-functional requirements.". URL <https://x.com/GergelyOrosz/status/2036925672501715425?s=20>. X.
- [69] OpenClaw. OpenClaw - Personal AI Assistant. URL <https://openclaw.ai/>. OpenClaw.
- [70] Claude API Docs. Adaptive thinking. URL <https://platform.claude.com/docs/en/build-with-claude/adaptive-thinking>. Claude API Docs.
- [71] Russell Coleman. Eval awareness in Claude Opus 4.6's BrowseComp performance, 2026. URL <https://www.anthropic.com/engineering/eval-awareness-browsecomp>. Anthropic.
- [72] 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/>.
- [73] 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.
- [74] Apple. Security Overview. URL https://developer.apple.com/library/archive/documentation/Security/Conceptual/Security_Overview/Architecture/Architecture.html. Apple.

- [75] 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.
- [76] Viacheslav Potoropin. Hello world does not compile · Issue #1 · anthropics/claude-c-compiler, 2026. URL <https://github.com/anthropics/claude-c-compiler/issues/1>. GitHub.
- [77] 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.
- [78] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. Non-Functional Requirements in Software Engineering, 1999. URL <https://personal.utdallas.edu/~chung/B00K/book.html>. Kluwer Academic Publishing.