AGENTHORIZON: EVALUATING AGENTIC JUDGES FOR LONG-HORIZON COMPUTER-USE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Computer-use agents are capable of completing complex tasks, increasing the use of automatic judges to determine success, either for training or for evaluation without human involvement. Despite their flexibility, their reliability on long tasks spanning multiple applications remains unclear. A trajectory, composed of long sequences of screenshots and actions, may appear complete, but in reality violates constraints from the instruction or introduces an unwanted side effect. To identify these errors, a judge needs to carefully examine the trajectory with respect to the user's instruction. To this end, we introduce AGENTHORIZON, a benchmark of 1,373 computer-use tasks (instruction–trajectory pairs) drawn from 166 hours of human-recorded trajectories spanning three operating systems. By recording trajectories for closely related instructions, we can construct negative tasks by simply *swapping* the instructions. This paired design evaluates judges on their ability to distinguish a truly successful trajectory from one that completed a similar (but incompatible) request. We release the benchmark under three splits: a frontier split, AGENTHORIZON (AH), a simplified split, AGENTHORIZON-SIMPLE (AH-S), and a development split, AGENTHORIZON-DEVELOPMENT (AH-D). We further evaluate eleven judges by (1) directly passing the full trajectory (with up to 300 screenshots and actions), and (2) using them as coding agents across five agent harnesses. We find that our best agentic judge, GPT-5.5, achieves 80.9% balanced accuracy on the AH subset. We find that tool-use improves certain models but results in worse performance for open-weight models, and that judges differ drastically in their ability to accept a valid trajectory and reject failed ones. Our findings highlight the need for judges that are capable of locating and verifying often hidden evidence that a task was properly completed inside long interaction histories. We release our benchmark and results publicly.

1 INTRODUCTION

Recent computer-use agents are becoming increasingly capable of carrying out complex tasks through graphical interfaces, from editing a spreadsheet to preparing a design across several desktop applications. However, measuring their progress requires determining if a sequence of actions, or *trajectory*, correctly fulfills user requests. For longer workflows, this may require verifying details scattered across hundreds of screenshots and action logs; we need to identify whether an item was properly selected, whether a sensitive file was shared with the intended recipients, or whether a design was created with the valid specifications.

Existing computer-use benchmarks require scripts validating each task separately; they verify properties such as the URL on the final page, a record or field in the database powering a website, or the contents of a file (Zhou et al., 2023; Koh et al., 2024; Drouin et al., 2024; Xie et al., 2024). Such verifications enable the evaluation to be deterministic and repeatable, but become difficult to specify if a task has several valid outcomes. For example, an agent can create a valid spreadsheet but the records are displayed in different orders; a design task could result in a logo that fulfills the request of the user without matching a reference image pixel for pixel. Experienced reviewers can decide case-by-case whether a task was successfully completed, but reviewing long trajectories is expensive.

As an alternative to expert reviewing, we can use LLMs to judge a trajectory directly based on its instruction. Prior works, such as AgentRewardBench (Lù et al., 2025), evaluate several judges

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

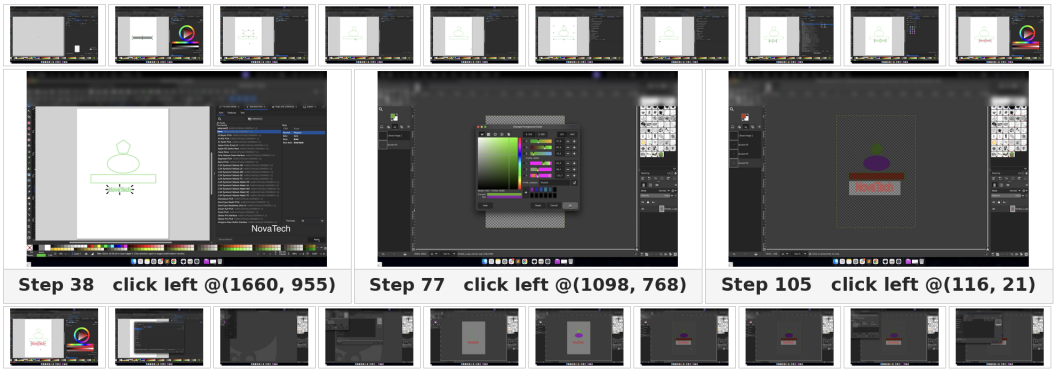


Figure 1: A plausible output can violate the instruction. In this 115-step Inkscape–GIMP recording (`0cacdac6`, macOS), a pentagon replaces the requested triangular accent. The mismatch survives recoloring and export, giving the crossed item a Critical Mistake label. The outer strips sample 10 frames from each half of the recording; the enlarged frames show the canvas, color selection, and final composite at steps 38, 77, and 105.

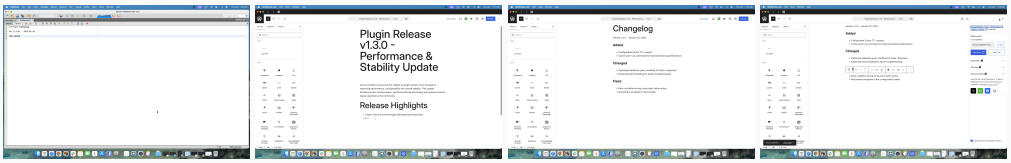
specialized for web agent trajectories. However, for long-horizon tasks on desktop, a judge must also determine which parts of the interaction history matter to claim a task was successfully completed. An LLM that receives the full representation of a task (henceforth, a *direct judge*) receives a fixed representation of that history in one request, whereas an *agentic judge* can use tools to filter and review any observation rather than ingesting the full context from scratch. However, prior work does not explore how reliably either approach identifies successful trajectories and whether the ability to inspect evidence through tools (using coding agents, for instance) can improve the quality and speed of the judgment.

Figure 1 highlights the importance of identifying precisely where an agent fails to follow the exact instruction for a task. For a task requiring Inkscape and GIMP, the instruction specifies that the finished logo design must contain a triangle. However, we observe in the 115-step trajectory that, somewhere along the way, a pentagon was drawn instead of a triangle, which deviates from the initial request. The subsequent recoloring and exporting continue as requested, giving the illusion that the full trajectory looks successful, which can only be uncovered if the judge verifies that the shape was drawn according to the request. Such errors may arise when a trajectory uses the wrong input, overlooks a constraint set in the instruction, or performs an action that was not originally requested. To evaluate such trajectories, we need to distinguish a trajectory that terminates properly (but may not respect the request) from a correctly completed task.

In order to evaluate whether a judge can correctly determine the success of a long-horizon computer-use trajectory with respect to the task instruction, we introduce AGENTHORIZON, a benchmark derived from human-recorded trajectories consisting of paired tasks. Our dataset requires annotators to produce two closely related instructions (i.e., a *match*) and record a trajectory for each of them. Pairing a trajectory recording with the instruction from its counterpart (i.e., *swapping*) always yields a negative example. Our benchmark consists of 523 positive and 850 negative trajectories, which we curate based on independent reviews of each trajectory. Its paired construction allows us to test whether a judge recognizes the difference in the instructions that separates a successful trajectory from an unsuccessful one, which differentiates our work from prior works focusing on trajectories that failed to produce any tasks (Lù et al., 2025; Li et al., 2026).

In order to enable the evaluation of a wider range of judges, we predefine three subsets. First, AgentHorizon (AH) contains the most challenging tasks (determined in Section 3.3) on which we test the frontier judges. Then, AgentHorizon-Simple (AH-S) tests judges that may be used for simpler tasks that may not yet push the frontier of long-horizon computer-use, but remain useful at several stages of model training. Finally, AgentHorizon-Development (AH-D) handles a subset focused on the development of judges specifically tuned or trained for computer-use tasks, avoiding repeated evaluation on the main splits.

Judging task (AGENTHORIZON negative; NetBeans + WordPress on macOS, 195 steps).
Instruction. In NetBeans, draft `plugin-release-1.3.0.md` (introduction + install snippet), then create a WordPress post assembled *from that release-notes file*, apply the `Development` category and four tags, preview, and publish.



Step 25. *NetBeans:* `changelog.md` *authored.* | **Step 85.** *WordPress editor: title and* Release Highlights *from the release-notes file.* | **Step 165.** *Same post now also carries a* Changelog *section (Added / Changed / Fixed).* | **Step 195.** *Post published; tags and category set.*

Judge prediction. Claude Opus 4.7 returns `success=true` with high confidence. *Reasoning (verbatim, abridged).* "Core objective satisfied: agent created both markdown files on Desktop in NetBeans, then opened WordPress and built a post with the required title, headings, lists, and code blocks assembled from the release notes. Parameters satisfied: category 'Development' and all four tags applied; post was previewed and then published."

Ground truth. `success=false`, mistake type *Bad Side Effect*. The instruction asked for a post assembled from the single `plugin-release-1.3.0.md` file. The agent additionally lifted the Added/Changed/Fixed sections from `changelog.md` and pushed them into the published post (step 165, surfaced again at step 195). The trajectory is a coherent successful execution of *this broader* release announcement, so the surface evidence the judge inspects is internally consistent. The judge accepts that consistency and misses the unrequested external publication that defines the Bad Side Effect category.

Figure 2: One AGENTHORIZON item: instruction, four illustrative keyframes (the LLM judge sees the full 195-step trajectory), judge verdict, and ground truth. Claude Opus 4.7 predicts `success=true`; ground truth is a Bad Side Effect (unrequested changelog content published).

When evaluated across eleven models and five agent harnesses, the strongest agentic judge achieves 80.9% balanced accuracy on AH. However, we found that access to tools does not consistently improve performance: whereas GPT-5.5 benefits from the Codex CLI over receiving the full context, Qwen 3.6 27B performs better with direct prompting than using it via OpenCode. We further find that judges differ in how they reject valid work or accept a task with mismatched instructions. Our results highlight limitations of current judges, and highlight the discrepancy between frontier models and strong open-weight models when it comes to judging tasks directly inside agent harnesses rather than direct judging.

2 RELATED WORK

Evaluating computer-use agents. WebArena (Zhou et al., 2023) and VisualWebArena (Koh et al., 2024) evaluate agents in self-hosted web environments. While WorkArena (Drouin et al., 2024) and WorkArena++ (Boisvert et al., 2024) study agents in compositional tasks of enterprise workflows, OSWorld (Xie et al., 2024) extends evaluation across operating systems. Expanding benchmarks to cover a broader range of tasks that agents can attempt requires examining the evaluators themselves. Specifically, Xue et al. (2025) and Zhu et al. (2025) identify errors and biases in existing evaluation methods. Furthermore, OS-Harm (Kuntz et al., 2025) evaluates agent safety with an automated judge validated against human annotations. In this work, we focus on evaluating the judges themselves against reviewed trajectory labels.

LLM judges. While LLM-based evaluation originally focused on generated responses of the agent, recent work extends it to the agent's behavior. Zheng et al. (2023) examine the position and verbosity biases of LLM-based judges, and Li et al. (2025a) provide a comprehensive survey of the broader evaluation paradigm. Additionally, RewardBench (Lambert et al., 2024) and RewardBench 2 (Malik et al., 2025) evaluate reward models on preference comparisons. For computer-use tasks, AgentRewardBench (Lù et al., 2025) contains 1,302 web-agent traces and shows that all tested judges achieve precision below 70% in determining the task success. While AgentRewardBench's failures originate in agent executions, our paired human demonstrations isolate discrepancies between a recorded workflow and its instruction. Agent evaluation surveys (Yehudai et al., 2025; Mohammadi et al., 2025) study trajectory assessment and discuss challenges with long-horizon trajectories, while Agent-as-a-Judge (Zhuge et al., 2024) proposes using tools to inspect generated workspaces.

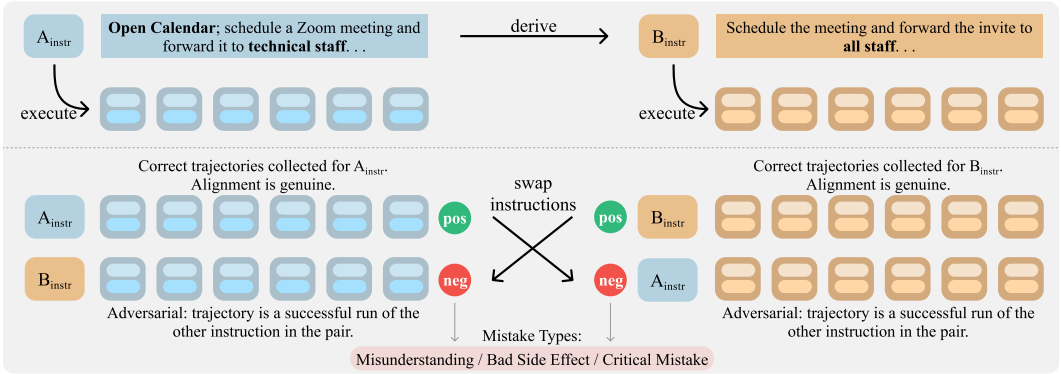


Figure 3: Paired construction before label review. Annotators record trajectories for related instructions A_{instr} and B_{instr}. Matched instruction–trajectory combinations are candidate positives; crossed combinations are candidate negatives. Positive retention and negative validity are then reviewed separately.

Trajectory and process evaluation. PaperBench (Starace et al., 2025) uses hierarchical rubrics to grade research replication, whereas SWE-PolyBench (Rashid et al., 2025) evaluates repository-level coding via execution-based evaluation. AssistantBench (Yoran et al., 2024) compares realistic web tasks against reference answers. Furthermore, while WebCanvas (Pan et al., 2024) and TheAgentCompany (Xu et al., 2024) grade intermediate trajectory steps or workspace outcomes, frameworks like OpenHands (Wang et al., 2025) and BrowserGym (de Chezelles et al., 2024) propose common agent and evaluation interfaces. Additionally, related work studies process supervision for mathematical reasoning (Lightman et al., 2023) and analyzes repeated-trial reliability in τ-bench (Yao et al., 2024). For multi-agent systems, literature studies the attribution of multi-agent failures to particular agents and steps (Zhang et al., 2025), and surveys multi-agent collaboration (Tran et al., 2025). Our focus in this work is specifically the judgment of complete, single-agent computer-use trajectories.

Judgments in learning pipelines. LLM-based judgments of trajectories determine which generated examples are used for training student models. Preference Leakage (Li et al., 2025b) identifies biases toward students related to the judge model. Other related approaches learn from generated behavior in different ways: SPIN (Chen et al., 2024) contrasts self-generated responses against human demonstrations, ARPO (Dong et al., 2025) applies reward-guided optimization to agents, and SkillWeaver (Zheng et al., 2025) distills experience into reusable skills. Relying on a judge for filtering successful trajectories exposes the learning pipeline to critical failure modes: discarding valid examples or admitting incorrect trajectories into training.

Adversarial evaluation and paired data. While contrast sets (Gardner et al., 2020) probe local decision boundaries and CheckList (Ribeiro et al., 2020) tests behavioral capabilities via templates and perturbations, counterfactually augmented data (Kaushik et al., 2019) leverages minimal label-flipping edits. Dynamic benchmarking frameworks like ANLI (Nie et al., 2019) and Dynabench (Kiela et al., 2021) collect adversarial examples iteratively, with platforms such as Dynaboard (Ma et al., 2021) facilitating evaluation as a service. Furthermore, TruthfulQA (Lin et al., 2022) targets imitative falsehoods, whereas confident learning (Northcutt et al., 2019) detects likely label errors in datasets. AgentHorizon adopts a paired design for constructing multimodal trajectories: the intervention modifies the instruction paired with a valid recording, rather than the recorded execution itself.

3 AgentHorizon Benchmark

Each item consists of an instruction, a sequence of screenshots, desktop actions, and a binary label indicating whether that execution satisfies the instruction. Negative items additionally carry a failure-type label. The recordings are performed by human annotators, and the agents evaluate these recordings rather than performing the tasks themselves.

Table 1: Composition of the three disjoint benchmark subsets. Mistake-type counts are over negatives; the six negatives without a typed label are shown separately. AH-D is reserved for development and excluded from both evaluation subsets.

	AH-D	**AH**	**AH-S**
Judging tasks	162	528	683
Positives	62	227	234
Adversarial negatives	100	301	449
Mistake types among negatives			
Critical Mistake	32	48	193
Bad Side Effect	26	107	99
Misunderstanding	41	142	156
Untyped	1	4	1

3.1 Task coverage

The dataset contains 166 hours of screen recordings, split across Windows, Linux, and macOS on six task domains. The tasks are completed across one or more applications spanning office software, design tools, and system utilities. We produce a screenshot at each time step and associate it with the corresponding action, such as a click, text entry, scroll, or application switch. The median item involves 121 steps, while 83 items involve at least 200 steps.

3.2 Paired construction and label review

We ask annotators to create two closely related instructions that differ in a requirement affecting task success and to record a demonstration for each of the instructions. Denote the instructions by A_{instr} and B_{instr}, and their recordings by A_{traj} and B_{traj}. The corresponding matched combinations $(A_{\text{instr}}, A_{\text{traj}})$ and $(B_{\text{instr}}, B_{\text{traj}})$ are positive candidates. Swapping the recordings gives the negative candidates $(A_{\text{instr}}, B_{\text{traj}})$ and $(B_{\text{instr}}, A_{\text{traj}})$ (Figure 3). The negative candidate example still shows a very coherent execution, but it is paired with a request that the execution does not satisfy.

We now remove the pairs with semantically equivalent instructions. This leaves 425 complete pairs, with two recordings per pair and four instruction–trajectory candidates per pair, for a total of 850 recordings and 1,700 candidates. Positive candidates undergo an initial quality review followed by an evidence review. After arbitration, 135 of the remaining 462 candidates are retained, while 327 are excluded because their concerns could not be resolved. Therefore the final positive set consists of 523 items.

The negative candidates are reviewed by checking whether the recording satisfies the swapped instruction. Removing a positive candidate does not necessarily require removing its paired negative. The negative remains valid if the recording does not satisfy the swapped instruction. The vendor reviewed the negatives from every pair, covering 850 items in 212.5 contributor-hours. All were retained, bringing the total reviewed set to 1,373 items, including 523 positives. Appendix A.1 summarizes the corresponding annotation and review record.

We group mismatches into three failure types. A *Critical Mistake* means that the trajectory fails to accomplish the primary objective. A *Bad Side Effect* occurs when the intended goal is completed, but introduces harm, risk, or unwanted consequences. A *Misunderstanding* follows a plausible but incorrect interpretation without such external harm. Table 5 states the adjudication rules. Figure 2 shows an example of how a judge can accept a complete trajectory despite an unrequested publication.

3.3 Development holdout and difficulty partition

We reserve the 162 retained items from the pair-aware development set as AgentHorizon-Development (AH-D) which includes 62 positives and 100 negatives. Their instruction pairs are excluded from the evaluation pools, leaving 1,211 examples. We estimate the difficulty of each remaining example using Qwen 3.5 122B-A10B, Inkling and Kimi K2.7 Code. Each model produces eight verdicts per item. An item is assigned to AgentHorizon (AH) if at most 18 of these verdicts

agree with the reference label. Otherwise, it is assigned to AGENTHORIZON-SIMPLE (AH-S). Any invalid model output is treated as an incorrect verdict.

This procedure gives 528 AH items (227 positive, 301 negative) and 683 AH-S items (234 positive, 449 negative). Evaluating the subsets shows whether a judge performs better on difficult examples without losing reliability on easier ones. Difficulty here is defined by the selected models and threshold, rather than assumed to be an inherent property of a task. We examine the sensitivity to these choices in Appendix A.3.

3.4 METRICS

We report balanced accuracy for AH and AH-S by averaging the accuracies on positive and negative examples. This provides both classes equal weight, such that a judge that always predicts the same label scores 50%. Each class uses its full denominator with missing, malformed, null, and non-Boolean predictions counted as errors. The scoring script matches each prediction to the versioned manifest using the task ID and rejects duplicate or unknown IDs. AH-D is excluded from both evaluation scores.

We use class-specific accuracy to examine the rejection of valid demonstrations from acceptance of mismatches. We also report the *mistake-type recall*, which counts the fraction of negatives in a category for which the judge predicts both failure and identifies the reference category. The MT% score combines these correct predictions over all negatives. Per-trajectory cost and runtime for each trajectory serve as measures of computational efficiency (Appendix B.1).

3.5 SUBSET COMPOSITION

Table 1 gives label and failure-type counts for all three subsets. Appendix A.5 reports trajectory length and application coverage, with corresponding fields in the row-level manifest for further analysis.

4 EXPERIMENTS

Judge models. We evaluate six closed-source models: Anthropic Claude Opus 4.7 (Anthropic, 2026), Anthropic Claude Haiku 4.5 (Anthropic, 2025), Google Gemini 3.1 Pro (Google Gemini Team, 2026b), Google Gemini 3.1 Flash Lite (Google Gemini Team, 2026a), OpenAI GPT-5.4 mini (OpenAI, 2026b), and OpenAI GPT-5.5 (OpenAI, 2026a). We also evaluate five open-weight models, which are Qwen 3.6 27B (Qwen Team, 2026b), Qwen 3.6 35B-A3B (Qwen Team, 2026c), Qwen 3.5 9B (Qwen Team, 2026a), Gemma 4 31B, and Gemma 4 26B-A4B (Farabet & Lacombe, 2026). We host open-weight models using vLLM[1] on H100 GPUs.

Our agentic judges use several harnesses which includes Claude Code, Codex, Gemini CLI, OpenHands and OpenCode. Each agent is tasked with inspecting the trajectory using tool calls and by reading screenshots selectively. These harnesses differ in how they select screenshots, manage context and expose tools. We also evaluate nine models as direct judges, giving them a fixed screenshot representation in one chat-completion call. Both approaches return a success verdict, confidence, a failure type when applicable, and a rationale. Table 2 reports the agentic judge and direct judge results, and Appendix B.3 describes the additional model-harness pairings.

4.1 JUDGE PERFORMANCE

We find that GPT-5.5 with Codex harness achieves the best balanced accuracy at 80.9%, followed by Gemini 3.1 Pro with Gemini CLI (77.7%) and Claude Opus 4.7 with Claude Code (76.0%). Qwen 3.6 27B with OpenCode is the strongest open-weight agentic judge at 70.4%. Most systems perform substantially better on AH-S, as expected for the easier subset.

We observe that the benefit of using agentic judging changes with models. Qwen 3.6 27B reaches 77.6% with direct judgment, exceeding its OpenCode based result by 7.2 points. On the other hand, GPT-5.5 shows the reverse pattern by improving from 68.6% with direct prompting to 80.9% with

[1] https://vllm.ai/

Table 2: Judge performance on the held-out evaluation subsets. AGENTHORIZON contains the 528 items that remain difficult under the pooled three-model split; AGENTHORIZON-SIMPLE is its 683-item complement. Pos and Neg are class-specific accuracies on AH, and Bal. is their mean. * marks direct rows using the 2×2 image-merge preprocessing described in Appendix B.1.

Model	Open?	Interface	AH Bal.	AH Pos	AH Neg	AH-S Bal.
Agentic judges						
GPT-5.5		Codex	**80.9**	71.8	90.0	92.6
Gemini 3.1 Pro		Gemini CLI	77.7	65.6	89.7	86.3
Claude Opus 4.7		Claude Code	76.0	76.7	75.4	94.6
Qwen 3.6 27B	✓	OpenCode	70.4	75.3	65.4	93.6
GPT-5.4 mini		Codex	69.2	67.4	71.1	91.2
Qwen 3.6 35B-A3B	✓	OpenCode	64.6	77.1	52.2	90.5
Claude Haiku 4.5		Claude Code	52.2	43.2	61.1	87.7
Gemini 3.1 Flash Lite		Gemini CLI	52.1	58.6	45.5	75.2
Gemma 4 26B-A4B	✓	OpenCode	49.4	70.5	28.2	69.9
Gemma 4 31B	✓	OpenCode	46.8	61.7	31.9	87.5
Direct LLM judges						
Qwen 3.6 27B*	✓	Direct	**77.6**	93.8	61.5	96.0
Gemini 3.1 Pro*		Direct	77.4	70.0	84.7	88.4
Qwen 3.5 9B*	✓	Direct	70.3	91.2	49.5	91.9
Qwen 3.6 35B-A3B*	✓	Direct	69.5	95.2	43.9	92.7
GPT-5.5		Direct	68.6	56.4	80.7	84.4
GPT-5.4 mini		Direct	66.5	85.9	47.2	94.0
Gemma 4 31B	✓	Direct	60.1	63.0	57.1	86.6
Gemini 3.1 Flash Lite		Direct	53.6	64.8	42.5	75.3
Gemma 4 26B-A4B	✓	Direct	46.5	44.5	48.5	72.6

Codex. Thus, allowing a model to inspect the trajectory through tools does not necessarily improve its verdicts. However, these comparisons change the overall harness, so they do not isolate the effect of screenshot selection, context management, or any other individual component.

4.2 EXAMPLE FAILURE CASES

To illustrate how judges fail to predict the correct outcome, we examine three example judgments from Claude Opus 4.7.

Unrequested external action. In task `7eb6defc` (using Trello and Slack), the agent aimed to complete a project-management workflow requested by the user, followed by updating the status and posting to an organization-wide `#all-staff` channel. In this case, we matched the instruction with its counterpart trajectory, resulting in a bad side effect. The Claude Opus 4.7 judge classified the trajectory as successful, whereas GPT-5.5 rejected it. This highlights that verifying that the requested output exists is insufficient, especially when an additional action changes the recipient (in this case, sending the message to the wrong channel).

Classifying a successful trajectory as failure. The agentic judge powered by Claude Opus 4.7 marks task `6debd146` (using GitHub and VS Code) as a failure, claiming that a clone is stored in GitHub Desktop's default repository folder rather than the `Documents/repo` path named in the instruction. GPT-5.5 considers the trajectory to correctly solve the task instruction, which is backed by the ground truth label. This shows how judges may believe a deviation should affect the success or failure classification of a task.

5 DISCUSSION

Our results demonstrate that both selecting relevant evidence and finding what that evidence implies for the user's request contribute to evaluating a long trajectory. Experiments reveal that agentic judges can revisit screenshots and actions, but our direct judge results show that this flexibility

is not sufficient by itself. We observe that the choice of harness also matters: on the same AH items and with an identical prompt, swapping OpenCode for Codex alters balanced accuracy by 3.7 points for GPT-5.5 and 1.2 points for Qwen 3.6 27B, even though it preserves their order (Appendix B.3). Understanding these detailed differences requires investigating how judges select and process evidence and observations, rather than assuming that additional tool access improves the accuracy.

Our paired example design further clarifies why recognizing a completed workflow is insufficient. Qwen 3.6 27B classifies 98.9% of both matched and swapped full trajectories as successful without the corresponding instruction given as input (Appendix B.4). This demonstrates that the two classes seem similarly complete when the instruction is omitted from the input and success is undefined without the user's request. Even with the instruction provided, the Bad Side Effect failure type shows the lowest exact recall across all the primary judges on the full reviewed corpus (Appendix B.2). Consequently, a judge may decide that the requested output exists while overlooking a specific action that makes the trajectory unacceptable.

The consequences of these errors vary based on the judge and how it is used. Rejecting successful trajectories underestimates agent capabilities and removes valid and useful examples from training data; conversely, accepting invalid trajectories has the opposite impact. While balanced accuracy reflects both types of error without allowing the larger error-type class to dominate, it fails to assign them application-specific costs. As a result, the opposing positive and negative accuracies in Table 2 are relevant even when two judges achieve similar overall performance scores.

Lastly, the difficulty partition is useful only if the outcomes are not sensitive to the model that was used to define it. Our analysis shows that rankings across the three different model-defined AH subsets exhibit high correlation ($\rho = 0.963$–0.975) despite the variations across items assigned to each subset. The ensemble partition further offers a challenging set whose principal ranking trends persist across the individual splitters.

Limitations. Our paired design analysis focuses specifically on subtle near-miss errors rather than the full distribution of agent failures (e.g., loops, crashes, stalls, and gradual drift). Therefore, the instruction-free controls should not be used to model general agent failures. Since the difficulty split depends on the recorded splitter configurations, provider, and model revision, its alternative splits serve to evaluate the robustness across individual splitters rather than isolated model effects. Although AH-D excludes the pair-aware final development set, earlier pilot exposure constrains claims of completely unseen evaluation data. Furthermore, independent human inter-annotator agreement was not collected. Correlation between original and re-examined labels measures consistency across review stages, rather than agreement among independent annotators. Lastly, the corpus is restricted to English, covers only a finite set of applications across three operating systems, and offers only screenshots and action logs for evaluation rather than comprehensive application states.

Future directions. Evaluating common agent failures alongside paired mismatches establishes how well these empirical findings generalize to deployed agents. Moreover, comparing screenshot evidence selection and long context management strategies across agents within a single harness helps identify the reason behind the tools enhancing some judges but leaving others unaffected. Additionally, the provided recordings allow further direct evaluation from video and the instruction pairs provide training examples where modifying a specific subtle requirement changes whether a trajectory succeeds.

6 Conclusion

We introduce AgentHorizon, an evaluation suite targeting judge performance across long computer-use trajectories. By pairing human-recorded workflows with closely related fine-grained instructions, it probes whether judges identify subtle failures in executions that otherwise appear complete. Our evaluation of eleven judge models reveals that current judges still struggle with making this critical distinction, and that tool use does not consistently boost their performance. We hope that AgentHorizon will support the development of more robust, reliable judges for computer-use agents, emphasizing the detection of missed constraints and unintended side effects.

AI USE STATEMENT

We used coding agents to help with the experimental code, the initial draft, and figure and table preparation, but they were not used for recording the demonstrations, which followed a protocol prohibiting AI-assisted task creation. The content of the final manuscript was completely written by the authors. We also used AI tools for finding relevant literature for our related work, producing scripts for analyses, and assisting with experiment design, data processing and analysis, and result interpretation. We verified the full manuscript to ensure the results and citations are correct, and take responsibility for the final content.

ETHICS STATEMENT

We employed professional annotators for the demonstrations, and they followed a protocol that requires accounts and profiles to be created for the purpose of the annotation. The protocol also prohibits personal communications and authentication secrets (Appendix A.1). Our benchmark focuses on evaluating whether an agent adheres to an instruction, but does not consider the safety aspect (whether an agent is completing a task that's inherently dangerous), so a high score should not be considered sufficient to deploy an agent into production.

REPRODUCIBILITY STATEMENT

In Section 3.2 and Appendix A.1, we document the exact process of constructing and reviewing the trajectories in our benchmark, allowing future work to replicate our benchmark design. Section 3.3 and Appendix A.3 focus on how we achieved the three-way partitioning using strong open models, whereas Section 3.4 discusses scoring denominators and how to handle invalid outputs, which complements our open-source code for running the evaluation. In Appendix B.1, we describe the preprocessing and compute used to run our experiments in detail.

REFERENCES

Anthropic. Introducing Claude Haiku 4.5, October 2025. URL https://www.anthropic.com/news/claude-haiku-4-5.

Anthropic. Introducing Claude Opus 4.7, April 2026. URL https://www.anthropic.com/news/claude-opus-4-7.

L'eo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier de Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *ArXiv*, abs/2407.05291, 2024.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6621–6642. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chen24j.html.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Alexandre Drouin, Massimo Caccia, L'eo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *ArXiv*, abs/2412.05467, 2024.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *ArXiv*, abs/2507.19849, 2025.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, I. Laradji, Manuel Del Verme, Tom Marty, L'eo Boisvert, Megh Thakkar, Quentin Cappart, David Vázquez, Nicolas Chapados, and Alexandre

Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? *ArXiv*, abs/2403.07718, 2024.

Clement Farabet and Olivier Lacombe. Gemma 4: Byte for byte, the most capable open models, April 2026. URL https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating nlp models via contrast sets. *ArXiv*, abs/2004.02709, 2020.

Google Gemini Team. Gemini 3.1 Flash-Lite: Built for intelligence at scale, March 2026a. URL https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-flash-lite/.

Google Gemini Team. Gemini 3.1 Pro: A smarter model for your most complex tasks, February 2026b. URL https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/.

Divyansh Kaushik, E. Hovy, and Zachary Chase Lipton. Learning the difference that makes a difference with counterfactually-augmented data. *ArXiv*, abs/1909.12434, 2019.

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. *ArXiv*, abs/2104.14337, 2021.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Thomas Kuntz, Agatha Duzan, Haochen Zhao, Francesco Croce, Zico Kolter, Nicolas Flammarion, and Maksym Andriushchenko. Os-harm: A benchmark for measuring safety of computer use agents. *ArXiv*, abs/2506.14866, 2025.

Nathan Lambert, Valentina Pyatkin, Jacob Daniel Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Raghavi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hanna Hajishirzi. Rewardbench: Evaluating reward models for language modeling. *ArXiv*, abs/2403.13787, 2024.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of LLM-as-a-judge. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.138. URL https://aclanthology.org/2025.emnlp-main.138/.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in llm-as-a-judge. *ArXiv*, abs/2502.01534, 2025b.

Zehao Li, Zhenyu Wu, Yibo Zhao, Bowen Yang, Jingjing Xie, Zhaoyang Liu, Zhoumianze Liu, Kaiming Jin, Jianze Liang, Zonglin Li, Feng Wu, Bowen Zhou, Zun Wang, and Zichen Ding. Os-themis: A scalable critic framework for generalist gui rewards, 2026. URL https://arxiv.org/abs/2603.19191.

H. Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher Pal, and Siva Reddy. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *ArXiv*, abs/2504.08942, 2025.

Zhiyi Ma, Kawin Ethayarajh, Tristan Thrush, Somya Jain, Ledell Wu, Robin Jia, Christopher Potts, Adina Williams, and Douwe Kiela. Dynaboard: An evaluation-as-a-service platform for holistic next-generation benchmarking. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 10351–10367. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Daniel Morrison, Noah A. Smith, Hanna Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. *ArXiv*, abs/2506.01937, 2025.

Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 6129–6139. ACM, August 2025. doi: 10.1145/3711896.3736570. URL http://dx.doi.org/10.1145/3711896.3736570.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, J. Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. *ArXiv*, abs/1910.14599, 2019.

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *J. Artif. Intell. Res.*, 70:1373–1411, 2019.

OpenAI. Introducing GPT-5.5, April 2026a. URL https://openai.com/index/introducing-gpt-5-5/.

OpenAI. Introducing GPT-5.4 mini and nano, March 2026b. URL https://openai.com/index/introducing-gpt-5-4-mini-and-nano/.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bingqian Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. Webcanvas: Benchmarking web agents in online environments. *ArXiv*, abs/2406.12373, 2024.

Qwen Team. Qwen3.5-9B model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-9B.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026c. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Muhammad Shihab Rashid, Christian Bock, Yuan Zhuang, Alexander Buchholz, Tim Esler, Simon Valentin, Luca Franceschi, Martin Wistuba, Prabhu Teja Sivaprasad, Woo Jung Kim, Anoop Deoras, Giovanni Zappella, and Laurent Callot. Swe-polybench: A multi-language benchmark for repository level evaluation of coding agents. *ArXiv*, abs/2504.08703, 2025.

Marco Tulio Ribeiro, Tongshuang Sherry Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *ArXiv*, abs/2005.04118, 2020.

Giulio Starace, Oliver Jaffe, Dane Sherburn, J. Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, E. Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate ai research. *ArXiv*, abs/2504.01848, 2025.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *ArXiv*, abs/2501.06322, 2025.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, T. Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *ArXiv*, abs/2404.07972, 2024.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Meng Bao, Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Ming-Hsuan Yang, Hao Lu, Amaad Martin, Zhe Su, L. Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *ArXiv*, abs/2412.14161, 2024.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, D. Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *ArXiv*, abs/2504.01382, 2025.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *ArXiv*, abs/2406.12045, 2024.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents. *ArXiv*, abs/2503.16416, 2025.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. AssistantBench: Can web agents solve realistic and time-consuming tasks? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8938–8968, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.505. URL https://aclanthology.org/2024.emnlp-main.505/.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *ArXiv*, abs/2505.00212, 2025.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Z. Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *ArXiv*, abs/2504.07079, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, E. Xing, Haotong Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685, 2023.

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

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy K. Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, J. Dhamala, Jacob Merizian, Mario Giulianelli, H. Coppock, C. Ududec, Jasjeet Sekhon, Jacob Steinhardt,

A. Kellerman, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices for building rigorous agentic benchmarks. *ArXiv*, abs/2507.02825, 2025.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jurgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. *ArXiv*, abs/2410.10934, 2024.

A BENCHMARK DETAILS

A.1 ANNOTATION PROCESS DETAILS

For the annotation process, professional annotators were tasked with designing task pairs and recording long trajectories on one of the three approved operating systems (Windows, macOS, Ubuntu), with the usage of AI being prohibited. Each action they perform is stored as a structured record alongside the screenshots. Additionally, each trajectory was also recorded using OBS, yielding 166 hours of high-resolution recordings, using 11.1 GB of storage.

Pair construction The tasks were designed as minimally different pairs. Starting from a seed task, a second (i.e., "child") task is derived such that accidentally completing the child task when instructed to complete the parent task would result in a failure that can be categorized as one of the following: misunderstanding, bad side effect, and critical mistake (see Table 5). Conversely, an agent that completes the parent task when asked to complete the child task would also be considered a failure, which is independently categorized within the same three categories; although the error type might often be the same, it may sometimes differ as they do not have a necessarily symmetric relation. Finally, the annotators are asked to complete **both** tasks independently, using the same subset of software.

Table 3: Vendor-reported effort over the 850 unique trajectories. The delivered total exceeds the original 1,960-hour estimate because calibration introduced additional review passes.

Stage	Hours/task	Total hours
Annotation	4.00	3,400.0
Review L1	1.70	1,445.0
Mistake-category review	0.25	212.5
Second-stage review	1.20	1,020.0
Total	**7.15**	**6,077.5**

A.2 MISTAKE TAXONOMY AND CALIBRATION RUBRIC

Our three labels show all the ways a trajectory would be considered a failure to complete the provided instruction. We provide a calibration document, *Mistake Type Definitions and Evaluation Criteria*, to the annotators to ensure alignment among the annotation team; it is included with the supplementary artifacts. It includes supplementary instructions about the concepts discussed in Table 5, such as a decision flow, pair-level examples, and a recovery-cost matrix.

A.3 PARTITIONING MODEL AND SUBSET DEFINITION

Our three splits are all grouped by instruction pairs to prevent information leakage across datasets. Our AH-D subset contains 162 trajectories, with the remaining 1,211 distributed across the other splits. For the remaining items, we decided to use an ensemble of three strong open models to identify truly challenging tasks, making the process reproducible: Qwen 3.5 122B-A10B, Inkling, and Kimi K2.7 Code. Each model is used as an agentic judge to produce 8 repeated judgments for each item (in order to ensure consistency), leading to a total of 24 judgments. We assign items with at most 18 matches with the ground truth to AGENTHORIZON, and the remaining items are assigned to AGENTHORIZON-SIMPLE.

Table 4: Stage-by-stage benchmark accounting. The 327 removals affect candidate positives only; AH-D, AH, and AH-S are disjoint and exhaust the reviewed benchmark.

Stage	Positive	Negative	Total
Paired construction pool	850	850	1,700
Reviewed benchmark	523	850	1,373
AH-D	62	100	162
AH	227	301	528
AH-S	234	449	683

Table 5: Operational boundaries of the annotator-facing failure taxonomy.

Type	Boundary	Typical recovery
Critical Mistake	The core action or target is not achieved.	Restart or repeat the task; the original outcome cannot satisfy the request.
Bad Side Effect	The goal is achieved, but an unrequested action creates cost, risk, exposure, or cleanup.	Reverse an external action, coordinate with others, or spend substantial effort undoing it.
Misunderstanding	The execution is coherent but misreads a parameter or constraint without external harm.	A local edit or short correction is sufficient.

A.4 Prompt development

We develop prompts iteratively on the development set (AH-D) by examining problems that may arise, including the lack of tool use or screenshot usage by the model. Our final prompt is presented in Figure 4.

```
You are judging one AgentHorizon trajectory. The judging framework, file layout, and mistake-
     type definitions are in AGENTS.md / CLAUDE.md / GEMINI.md in your working directory; use
     them.

# Input

Trajectory ID: `{{TRAJECTORY_ID}}`

The trajectory content is NOT in your context. You MUST read it from disk.

Start by opening `./agenthorizon_md/{{TRAJECTORY_ID}}.md` to get the goal and action sequence.
      Open the structured JSON at `./agenthorizon_json/{{TRAJECTORY_ID}}.json` or the
     screenshots when the markdown is ambiguous. One-shot guesses without reading the
     trajectory are not acceptable.

## Screenshots

Each step in the markdown ends with a screenshot link in the form
`![Step N screenshot](./data/media/images/<deliverable_id>/step_<N>.png)`. These are **local
     relative paths** in your working directory; use the `Read` tool (or your harness's
     equivalent) on the path directly to view the image. Do not attempt to fetch the
     screenshot from any external URL -- the local file is the source of truth and the model
     can view it inline.

Please look at least one screenshot.

# Output

Respond with a JSON object:
{
  "success": true or false,
  "reasoning": "One to three sentences citing which layer (core objective, parameters, or
     constraints) was or was not satisfied.",
  "confidence": "low" | "medium" | "high",
  "mistake_type": null when success=true, else one of:
    "Critical Mistake", "Bad Side Effect", "Misunderstanding of the Instruction"
}

Notes: confidence is evidence strength, not verdict. Low confidence ≠ failure. mistake_type is
      null on success.
```

Figure 4: Final evaluation prompt.

Table 6: Resource use for the primary agentic-judge configurations. Means are computed over all 1,373 trajectories unless a harness omitted the corresponding telemetry; self-hosted rows have no marginal API charge.

Model	Interface	Input tokens	Tools	Images	$/trajectory
GPT-5.5	Codex	513,556	18.6	11.9	0.4035
Gemini 3.1 Pro	Gemini CLI	1,393,271	24.2	30.2	0.6172
Claude Opus 4.7	Claude Code	554,387	9.2	7.8	0.6592
Qwen 3.6 27B	OpenCode	243,440	8.1	6.0	0
GPT-5.4 mini	Codex	323,808	20.0	8.4	0.1066
Qwen 3.6 35B-A3B	OpenCode	190,550	8.1	5.6	0
Claude Haiku 4.5	Claude Code	452,266	10.6	8.3	0.1216
Gemini 3.1 Flash Lite	Gemini CLI	1,041,455	8.1	3.6	0.0949
Gemma 4 26B-A4B	OpenCode	243,892	3.9	1.2	0
Gemma 4 31B	OpenCode	101,528	2.3	1.0	0

A.5 Additional dataset statistics

In Figure 5, we show the distribution of trajectory step counts: the trajectories have a median of 121 steps, a 90th percentile of 186, and a maximum of 358; 83 of the data points have at least 200 steps.

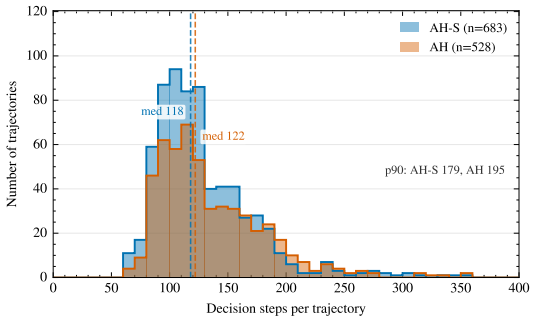


Figure 5: Decision-step distributions for the evaluation subsets. AH has a median of 122 steps and 90th percentile of 195; AH-S has 118 and 179, respectively.

Desktop application statistics Our benchmark covers the use of 112 desktop applications, with a mean 1.98 per task and a median of 2. 83.2% of tasks use exactly two applications, indicating that multiple apps are needed to complete long-horizon tasks.

B Supplementary results

B.1 Compute and preprocessing

Inference We use the official APIs of proprietary models, and run open-weight judges via vLLM on H100 GPUs.

Input representation For the direct judge approach, we feed one screenshot and action for each of the steps in a trajectory; in order to ensure reasonable context length, we downscale each screenshot to 512×332 pixels. For open-weight models that cannot support longer contexts, four consecutive frames are merged into a 2×2 mosaic. Our release pins the environment versions and provides the exact preprocessing and evaluation scripts to reproduce the experiments.

B.2 Mistake-type recall

To supplement results from our binary evaluations, we also include the fine-grained failure-type recall in Table 7. We found that Bad Side Effect has the lowest recall, indicating models are poorly calibrated to distinguish side effects from other mistake types.

Table 7: Exact mistake-type recall on the full benchmark for the primary agentic-judge configurations. Denominators are 273 Critical Mistakes, 232 Bad Side Effects, and 339 Misunderstandings; malformed outputs count as incorrect.

Model	Interface	Critical	Side effect	Misunderstanding
GPT-5.5	Codex	77.7	24.1	72.9
Gemini 3.1 Pro	Gemini CLI	57.5	7.3	87.3
Claude Opus 4.7	Claude Code	75.8	12.1	71.4
Qwen 3.6 27B	OpenCode	60.4	5.6	74.3
GPT-5.4 mini	Codex	60.8	2.6	74.6
Qwen 3.6 35B-A3B	OpenCode	63.0	6.0	53.1
Claude Haiku 4.5	Claude Code	58.6	8.6	53.4
Gemini 3.1 Flash Lite	Gemini CLI	62.3	1.7	29.2
Gemma 4 26B-A4B	OpenCode	47.6	0.0	18.6
Gemma 4 31B	OpenCode	74.0	0.0	27.1

Table 8: Models evaluated under multiple agentic interfaces on the final AH and AH-S splits. All runs use prompt P9 and the same benchmark inputs. AH Pos and AH Neg use fixed denominators of 227 and 301; AH-S Bal. uses 234 positives and 449 negatives. Missing or malformed outputs count as incorrect. † marks the primary interface reported in Table 2.

Model	Interface	AH Bal.	AH Pos	AH Neg	AH-S Bal.
GPT-5.5	Codex†	80.9	71.8	90.0	92.6
	OpenCode	77.2	70.9	83.4	92.2
Qwen 3.6 27B	Codex	71.6	70.5	72.8	94.5
	OpenCode†	70.4	75.3	65.4	93.6
	OpenHands	51.2	58.1	44.2	89.0
Gemini 3.1 Flash Lite	Gemini CLI†	52.1	58.6	45.5	75.2
	OpenCode	51.3	41.9	60.8	80.5
	Codex	47.8	38.8	56.8	73.1
	OpenHands	46.1	25.1	67.1	70.9

B.3 Evaluation with additional harnesses

In Table 8, we report every model that was evaluated with more than one agent harness, with GPT-5.5 and Qwen 3.6 27B being evaluated on both Codex and OpenCode, allowing a fair comparison. We found that GPT-5.5's performance is 3.7 percentage points lower when switching from Codex to OpenCode, while Qwen achieves 1.2 percentage points lower balanced accuracy on OpenCode and 20.4 percentage points lower on OpenHands. Based on those results, we can conclude that the choice of harness can substantially affect the performance of the judge, and encourage the use of native interfaces (for example, Codex for the GPT family), or otherwise several open harnesses to understand which one is a better fit.

B.4 Effect of removing required information

We study the effect of removing the instruction from the input passed to the judge, which would make this task impossible (since both trajectories should appear valid) in Table 9. We found that removing the instruction and action log results in balanced accuracy close to random (47.8% to 49.8%), whereas removing screenshots results in very poor performance (57.7%). This highlights that the trajectories by themselves do not provide enough information to predict the success, and that screenshots provide important information to make a proper judgment.

Table 9: Ablation results evaluating which trajectory components are necessary for judging a trajectory.

Instruction	Screenshots	Action log	TP	FN	FP	TN	BA
✗	✓ all	✓	246	5	273	0	**49.0%**
✓	— final only	✗	104	147	118	155	**49.1%**
✗	— final only	✗	215	35	245	26	**47.8%**
✓	✗	✓	205	46	181	92	**57.7%**
✗	✗	✓	249	2	272	1	**49.8%**