DELIVERBENCH: CAN MULTIMODAL AGENTS TURN BATCHES OF REAL-WORLD MATERIAL INTO USABLE DELIVERABLES?

Anonymous authors
Paper under double-blind review

ABSTRACT

Much of the work delegated to multimodal agents turns a batch of heterogeneous material into a structured artifact that downstream software consumes, where a single mishandled item can make the artifact unusable. We introduce DeliverBench, 129 tasks that each supply a batch of authentic material, including phone screenshots, photographed invoices and nameplates, spreadsheets, audio, and video, and require such a deliverable. Deliverables are verified item by item against 411 rubric criteria under a fixed set of 37 general tools, with every call logged. Across 27 frontier models, the strongest agent completes only **41.1%** of tasks, while human annotators scored by the same judge reach **87.0%**; **53 tasks are solved by no model**. The main empirical reversal is a **batch–horizon inversion**. Batch-processing tasks succeed at 16.0%, while long-horizon interactive tasks, commonly treated as the frontier difficulty, succeed at 46.0%. We hypothesize that interaction helps because it exposes verifiable intermediate states and clear stopping conditions; batch work offers neither, so one inconsistent item can silently invalidate the artifact. Two further findings concern how agents are measured. Countable efficiency summaries, such as tool calls, tokens, steps, and error rate, show no significant model-level association with success and should not be read as proxies for capability; a rubric-based rating of how each task was attempted does track success ($r = +0.677$). Tool-use strategy varies 19.6-fold and clusters strongly by vendor, yet is not significantly associated with success. Six shortcut screens and a paired cross-harness control support the validity of these measurements. We release the task pack, the rubric criteria, and the evaluation logs needed to reproduce every reported number.

1 INTRODUCTION

A large share of the work now delegated to agents consists of converting a batch of heterogeneous source material into a structured artifact that downstream software consumes: receipt photographs into an expense ledger, support screenshots into a triaged queue, nameplate photographs into an equipment register. Such tasks are seldom conceptually difficult. What makes them demanding is that every item must be read correctly and handled under one convention, and a single mishandled item makes the artifact unusable without anything in the environment signalling the error.

Existing agent benchmarks leave this setting largely unmeasured. Their terminal objects are a short answer, a tool plan, a trajectory, or an environment state produced from a single query (Mialon et al., 2024; Zhou et al., 2024; Xie et al., 2024; Guo et al., 2025; Zhou et al., 2025); none of these exposes whether each of many items was handled correctly and consistently. Tool surfaces, meanwhile, are either task-specific or very large and heterogeneous (Zhu et al., 2026; Liu et al., 2026; Bandi et al., 2026), so success partly reflects the instrument as well as the model. The agent loop that produces the score is also seldom held fixed: the same backbone ranges from 15.7% to 34.0% on CocoaBench across four scaffolds (CocoaBench Team et al., 2026), and vendor command-line harnesses do not surface failed calls the way an instrumented loop does, so neither outcome nor process measurements transfer across harnesses without an explicit control.

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.

DeliverBench targets this setting directly. Each of its 129 tasks places a model in a sandbox with a batch of authentic material, including phone screenshots, photographs of invoices and nameplates, spreadsheets, audio, and video, and requires a structured deliverable. Its design responds to each of these gaps. The deliverable is verified item by item against 411 rubric criteria, each routed to a judge that can observe its evidence: code assertions, LLM adjudication, file-content adjudication, or vision comparison (Section 3.5). The tool surface is fixed at 37 general tools, none written for a particular task, and every call is logged (Section 4). Process is scored on a track separate from the outcome, and a paired cross-harness control tests whether scores depend on the loop (Appendix B). We evaluate 27 frontier models, yielding 3,414 valid outcomes at 98.0% grid coverage. Figure 1 gives an overview.

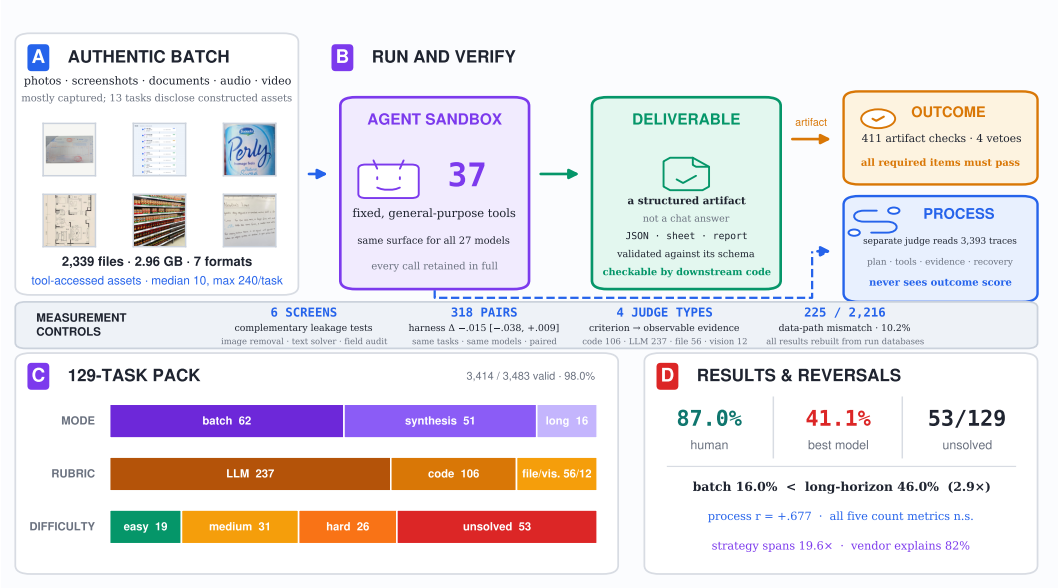

Figure 1: **DeliverBench at a glance.** **(A)** Source assets, captured or openly constructed, which the agent reaches only through tools. **(B)** A run must turn those assets into a program-checkable deliverable, with outcome and process scored on separate tracks. **(C)** The composition of the 129-task pack. **(D)** Headline results.

The results show a wide gap and an unexpected ordering of difficulty. The strongest model, Claude Opus 5 Max, delivers an acceptable artifact on 41.1% of tasks, against 87.0% for human annotators under the same judge, and 53 tasks are solved by no model. Batch-processing tasks succeed at 16.0%, whereas long-horizon interactive tasks, often regarded as the harder setting, succeed at 46.0%, which points to consistency across items, rather than interaction depth, as the main difficulty in this pack (Section 6.3). None of five countable trajectory summaries, such as tool calls, tokens, or error rate, is significantly associated with success at the model level, whereas a rubric rating of how the task was attempted correlates at r = +0.677. Tool-use strategy varies 19.6-fold and vendor identity explains 82% of its variance, yet it is not significantly associated with success.

Our contributions are as follows.

- **A benchmark for batch-to-artifact work.** 129 tasks over 2,339 authentic source assets (2.96 GB), each requiring a structured deliverable that is verified item by item by 411 criteria under a fixed, general 37-tool surface.
- **An evaluation of 27 frontier models with separate outcome and process tracks**, showing that models fail most on batch work rather than on long-horizon interaction, and that countable efficiency summaries are not proxies for capability.
- **Validity controls and a complete release**: a paired cross-harness control, under which vendor harnesses shift scores by a pooled −0.015 (95% CI [−0.038, +0.009]), six shortcut screens (Section 3.2), and the task pack, criteria, per-call logs, and scripts that reproduce every reported number.

2 Related Work

Multimodal tool use. VTC-Bench (Zhu et al., 2026) and VisualToolBench (Guo et al., 2025) chain image operators, GTA pairs real images with deployed tools (Wang et al., 2024a), m&m's evaluates multi-step multimodal tool plans (Ma et al., 2024), and AgentVista requires long-horizon hybrid tool use over detailed visual scenes (Su et al., 2026); M3-Bench aligns multimodal MCP tool calls against reference trajectories (Zhou et al., 2025). Each task poses one query and terminates in an answer, a plan, or a trajectory, as do the text-rich image benchmarks beneath them (Mishra et al., 2019; Kahou et al., 2017; Yue et al., 2024; Fu et al., 2025; Huang et al., 2026).

Tool orchestration and workflow composition. A second line tests whether agents can select and sequence many tools. MCP-Atlas composes cross-server workflows over 220 production tools and scores final answers against atomic claims (Bandi et al., 2026); Tool Decathlon spans workspace files and software state (Li et al., 2026); AppWorld exposes 457 APIs across nine apps and checks the resulting state with unit tests (Trivedi et al., 2024). OfficeBench, TheAgentCompany, and Spider2-V extend composition to cross-application office work, a simulated software company, and enterprise data workflows (Wang et al., 2024b; Xu et al., 2024; Cao et al., 2024). CocoaBench requires composing vision, search, and coding, and checks the final output with a per-task script (Hao et al., 2025; CocoaBench Team et al., 2026). TOBench is closest in modality and endpoint, with 324 tools over 27 MCP servers and omni-modal tasks that include artifact creation (Liu et al., 2026). In these benchmarks the difficulty lies in choosing and ordering tools; the material a task supplies is usually a single scenario rather than a batch of items that must each be read and handled under one convention.

Verifiable endpoints. Executable benchmarks check patches, terminal states, or GUI states (Jimenez et al., 2024; Deng et al., 2025; Merrill et al., 2026; Xie et al., 2024; Zhou et al., 2024), SheetCopilot checks spreadsheet states (Li et al., 2023), and GDPval uses blinded experts to grade occupational deliverables (Patwardhan et al., 2025). Conversational benchmarks place a simulated user in the loop (Yao et al., 2025; Barres et al., 2025). Table 1 compares the closest multimodal and orchestration benchmarks with DeliverBench.

Table 1: Comparison with related agent benchmarks. Tool and task counts are as reported by each benchmark; — marks a count not reported in comparable form.

Benchmark	Input	Tools	Tasks	Endpoint	Verification
VTC-Bench	Images	32 OpenCV ops	680	Answer	Reference answer
VisualToolBench	Images	6 general	1,204	Answer	Per-task rubric
GTA	Real images	Deployed tools	229	Answer	Step and answer accuracy
AgentVista	Images	4	209	Answer	Answer accuracy
M3-Bench	Images	231 (28 MCP)	—	Trajectory, answer	Call alignment, LLM judges
MCP-Atlas	Text	220 (36 MCP)	1,000	Answer	Claim-level rubric
AppWorld	Text	457 APIs (9 apps)	750	App state	State-based unit tests
CocoaBench	Vision, web, code	Open (ref. agent 39)	153	Final output	Per-task script
TOBench	Omni-modal	324 (27 MCP)	100	Artifact	Task-specific evaluators
DeliverBench (ours)	Multi-item batches, 5 modalities	37 general, fixed	129	Structured artifact	411 item-level criteria, 4 judge types

2.1 Unaddressed gaps

Among the benchmarks surveyed here, none combines batches of authentic multi-item material, a fixed general tool surface, and item-level verification of a structured deliverable. Document-understanding benchmarks read text-rich images but terminate in a short answer (Mathew et al., 2021; 2022; Masry et al., 2022; Liu et al., 2024b); agent benchmarks terminate in an answer, a trajectory or an environment state (Mialon et al., 2024; Zhou et al., 2024; Xie et al., 2024; Liu et al., 2024a; Deng

et al., 2023). Document and operations work of this kind falls between the two, and neither measures its characteristic difficulty, consistency across items.

Scaffold dependence is established, but it is benchmark-specific. Scaffold choice moves scores substantially (Yang et al., 2024; Wang et al., 2025); on CocoaBench one backbone ranges from 15.7% to 34.0% across four scaffolds (CocoaBench Team et al., 2026). How strongly a given benchmark's scores depend on the harness therefore has to be measured on that benchmark, with tasks and judge held fixed. Appendix B reports this control for DeliverBench.

3 The Benchmark

3.1 Task format

Each task contains a `task.yaml` with the prompt, an `assets/` directory, and an `evaluator/` directory with the rubric and code checkers. Two tasks illustrate the shape. Seven photographs of appliance energy labels are to be sorted into grades and emitted as keyed JSON; eight groups of restaurant receipts (full receipt, itemised crop, payment-area crop, plus who attended) are to be split across attendees in a nine-column CSV whose per-row and per-case totals must close against the printed total to two decimal places. Both are laborious more than difficult, and each ends in an object a program can check, where a single missed item shows up as a failure. Figure 6 in Appendix D gives a third example in full: eight unboxing photographs yield twelve independently adjudicated claims, including the distinction between a part absent from the frame and a part known to be missing.

3.2 Source material

One rule governs the pack: every value a model must produce is obtainable only by reading the source material. To enforce it, the pack was screened before release by six complementary tests: remove the images; check for keys and values on the same row; submit non-image assets as if they were answers; scan vector and text formats; build a text-only solver; and manually review every output field in screenshot tasks. Worked cases in Appendix A show why no one test subsumes the others. The rule also limits what else a task may contain: non-image files are permitted in exactly three roles: rules and thresholds, external conditions not visible in any image, and filename indices. Master data, gold answers and generation scripts never enter the model-visible area. Vector formats are banned outright.

Captured and constructed assets. Most of the material is photographed rather than drawn. A price board is shot at an angle with glare across the lower third; an unboxing photo has the accessory half-hidden behind the manual; a receipt is creased where it was folded into a wallet. We left these defects in place, since reading through them is part of the task. Where obtaining such material would mean handling personal or financial records, we construct the asset instead, and thirteen tasks say so in their own prompt text. Constructed assets pass the same review as captured ones: values internally consistent, layouts matching how such documents are issued, and answers still derivable only by reading the image. The review also checks that no constructed asset is easier to read than its real counterpart.

3.3 Taxonomy by mode of work

We classify tasks by the structure of the work rather than by domain, because structure determines what a model has to be able to do. Every task belongs to exactly one of three modes. **Batch processing** (62 tasks) hands over tens of items (median 25, up to 240), each handled individually, where the difficulty is to cover every item exactly once under one convention. **Synthesis** (51) makes items constrain one another so that no single item yields the answer. **Long-horizon interaction** (16) requires acting, observing the result, and correcting.

3.4 Measured difficulty

Tasks fall into four difficulty bands. The results under each are: `unsolved`, 53 tasks, 0% perfect with a median task-level mean score of 0.644; `hard`, 26 tasks, 4–30% and 0.674; `medium`, 31 tasks, 35–67% and 0.824; `easy`, 19 tasks, 70–100% and 0.935. A second field separates two situations

the bands conflate. Fifteen tasks are `exacting`: perfect-score rate at or below 10% while the mean is 0.75 or better; models typically complete much of the work, while few or none achieve a perfect score. The most extreme reaches a mean of 0.875 with nobody perfect. The bands describe the released pack and are exogenous to any future model; Appendix F notes why they cannot be used to make claims about the 27 models evaluated here.

3.5 Verification and scoring

Every completed task is scored by two rubrics that never exchange information: one reads the delivered artifact, the other reads the trajectory. Rubric grading by a model judge is standard for open-ended outputs (Arora et al., 2025; Starace et al., 2025; Guo et al., 2025; Zheng et al., 2023), but judges are imperfect (Tan et al., 2025), so each criterion goes to a judge that can observe its evidence, and to code whenever a programmatic check exists (Zhou et al., 2023).

Outcome. The 411 weighted criteria (median 3 per task) are checked item by item against the deliverable: `code` (106) for exact numeric and structural comparison, `llm` (237) for open reasoning and evidence statements, `file_llm` (56) for spreadsheet deliverables, and `vision` (12) for layout reproduction; Appendix H gives the assignment rules. Four veto criteria are excluded from the weighted mean but zero the task when failed. The score is the weighted mean over non-veto criteria, in [0, 1]. **A task is passed only at a perfect score ($\geq$ 0.999)**, because these deliverables are consumed by programs and a ledger with one wrong row must be corrected before use. The mean rubric score is reported beside every pass rate as a measure of how close a run came; across models the two orderings agree at only Kendall $\tau = 0.719$ (Section 5.1).

Process. A separate rubric scores how the task was attempted, never whether it succeeded: tool selection 25, plan coherence 20, efficiency 25, use of observations and self-correction 20, and adherence to process constraints 10. The judge receives the trajectory compressed to one line per step, together with deterministic statistics (step count, per-tool calls, repeated actions, and tokens), so it is never asked to count. Because the two tracks are independent by construction, Section 6 can ask whether they are related.

4 Experimental Setup

Sandbox and tools. Each task runs in its own Docker container built from the task's compose file, with ordinary outbound network access and no dedicated search service. Attachments are copied into `/workspace/inputs`. The agent is offered **37 tools** in five families: browser 15, DOM 10, file 7, image 2 (`image_read`, `image_crop`), and execution 2 (`code_execute`, `shell_execute`), plus `task_complete`. `image_crop` reads the original full-resolution file rather than the downscaled copy in context, which is how a model resolves fine print. None of these tools is task-specific (Appendix C).

Run configuration. All main experiments use one profile: an iteration cap of 200, an unlimited token budget, temperature 0 (providers with adaptive thinking may ignore it), and no image eviction from context. Every native function-calling provider receives one lean, provider-neutral instruction instead of provider-specific scaffolding (Appendix C).

Models. Twenty-seven models from eleven vendors have at least 100 valid cells, with reasoning effort set to the highest tier each provider exposes. Claude Fable 5.1 Max is reported but held out of the ranking because its 24 missing cells are not random (Appendix F); the other 26 models form the leaderboard. Three further models fall below 100 valid cells and are excluded.

Judge and valid cells. All adjudication uses one judge model, `gpt-5.5-xhigh`, for every task and every evaluated model. A (model, task) cell enters the denominator only if the run terminated normally **and** adjudication produced a usable score. Crashed runs, failed judge calls, and inputs exceeding a model's per-request limit are recorded as invalid rather than scored as zero, which would charge an infrastructure failure to the model. Of 3,483 reachable cells, 3,414 are valid (98.0%), with per-model n between 105 and 129. Two intersection scopes align denominators exactly: the 91 tasks completed by all 27 models, and 104 after excluding the lowest-coverage model.

Reproduction. Every statistic is regenerated by a single script from a frozen snapshot that reads the 3,414 per-run databases directly and records the repository commit, admitted runsets, and per-cell row-selection rule (Appendix D.12).

5 RESULTS

We evaluate 27 models and a human baseline over 129 tasks, yielding 3,414 valid model outcomes at 98.0% coverage. The perfect-score rate is the share of tasks on which a model delivered a usable artifact; the mean rubric score beside it measures how close a run came when it did not. One of the 27, Claude Fable 5.1 Max, is scored on only 105 tasks and is excluded from ranking. The table shows the top ten; the full leaderboard and behavioural means are in Appendix D.13.

Table 2: Human performance and the ten highest-ranked models. *Mean* is mean rubric score, *Perfect %* is the pass rate at ≥ 0.999, and *Proc* is the independent trajectory score. *Calls* and *kTok* are mean tool calls and output tokens in thousands per task. The complete leaderboard is in Appendix D.13.

#	Model	n	Mean	Perfect %	Proc	Calls	kTok
—	**Human** ‡	129	**0.953**	**87.0**	—	—	—
1	Claude Opus 5 Max	129	0.804	**41.1**	0.808	59	45
2	DeepSeek-V4.1-Flash(max)	121	0.774	36.4	0.737	80	80
3	Claude Opus 4.8	128	0.784	35.9	0.830	51	62
4	GPT-6-Astra-Max	127	0.784	35.4	0.823	46	34
5	Claude Fable 5	127	0.742	35.4	0.808	49	39
6	GPT-5.5-xHigh	129	0.735	34.1	0.750	42	25
7	Kimi-K3-Thinking	127	0.760	33.9	0.790	43	33
8	GPT-5.6-Sol-Max	129	0.748	33.3	0.753	49	95
9	GLM-5.3-Flash-Max	128	0.755	32.8	0.785	49	29
10	Gemini-3.8-Flash	127	0.761	32.3	0.621	66	66

‡ The human baseline covers all 129 tasks and is scored by the identical pipeline (same judge, thresholds, and veto criteria), with nothing in the scoring path knowing whether a submission came from a person or a model. Coverage is complete on both sides, so no estimator or reweighting is involved.

People clear the bar on 87.0% of the pack; the strongest of 27 frontier models clears it on 41.1%. Because the same judge and thresholds accept most human deliverables, rubric strictness alone cannot explain the model gap. For context, not as controlled cross-benchmark comparisons, reported human/model rates are 92%/15% on GAIA (Mialon et al., 2024), 78%/14% on WebArena (Zhou et al., 2024), and 94%/41% on TOBench (Liu et al., 2026); the task distributions, agent scaffolds, and scoring protocols differ.

Mean scores show where the deficit lies. Humans average 0.953 against 0.804 for the best model, 0.148 apart, far less in proportion than the 45.9 points separating the pass rates, and no model reaches a mean that high. Models therefore get most of the way through many tasks, but the remaining rubric error is often enough to make the artifact unusable (Appendix D.5).

Fifty-three of 129 tasks are solved perfectly by no model; exactly one is solved by all (Figure 2). The human result also constrains how to interpret the unsolved set. People solved 112 tasks, and only 76 tasks were solved by any model, so at least 36 of those 53 (better than two thirds) were solved by a person. Much of this band is therefore solvable by humans and cannot be explained solely by task mis-specification.

`Proc` is the independent trajectory score of Section 3.5, which never learns whether the task succeeded. It spans 0.621 to 0.830, while the pass rate spans 32.5 points; their rankings also come apart at both ends. The highest process score belongs to Claude Opus 4.8 (0.830), third on outcome; the lowest to Gemini-3.8-Flash (0.621), tenth. Qwen3.8-Max-Thinking, last in the full leaderboard (Appendix D.13), earns 0.700 and exceeds seven models that beat its outcome by 7 to 24 points.

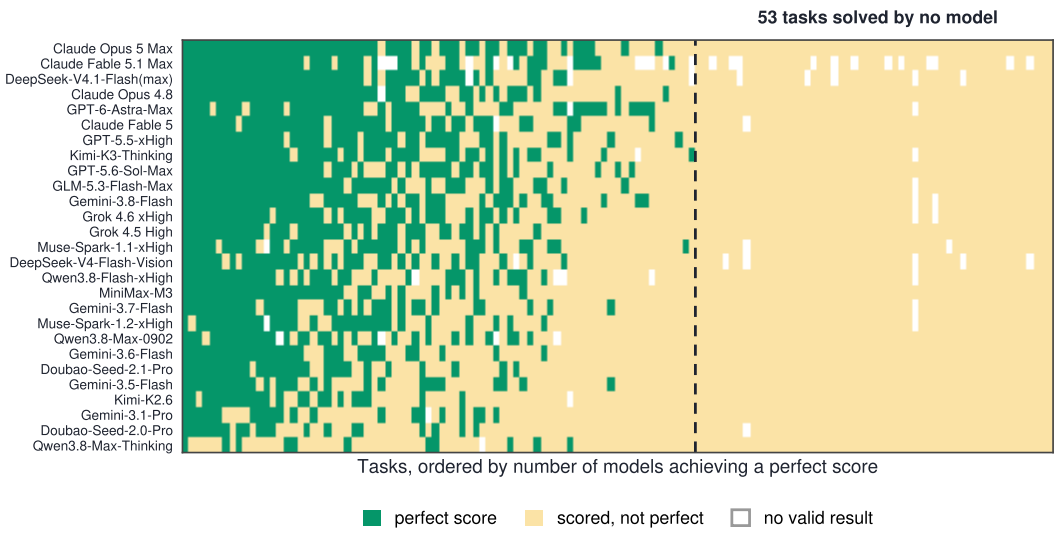

Figure 2: **Outcome matrix for 27 models and 129 tasks.** Each row is one of 27 frontier models and each column one of 129 tasks, ordered by how many models achieve a perfect score. The solid band on the right contains 53 tasks with no perfect model submission. Most cells in this band are scored-but-imperfect rather than missing.

5.1 Robustness of the ranking

Aggregation changes the ranking. Mean score and perfect-score rate agree at only Kendall τ = 0.719 (p = 1.5e-7) with 3 of 27 ranks identical, and for specific pairs they return different significance classifications (Appendix D.5). Which one a benchmark reports therefore affects which model a reader concludes is better.

Task-set scope is stable. The obvious worry about a grid with gaps is that a model looks good by having skipped the hard tasks. Restricting to the 91 tasks all 27 models completed leaves the ordering essentially untouched (τ = 0.864, p = 4.6e-10), giving no evidence that skipped hard tasks drive the ordering.

Resolution. Adjacent releases separated by less than roughly 7 points of pass rate are not distinguishable here, and the smallest difference we do detect is 14 points (Appendix D). Gaps in the table below this resolution should not be interpreted.

6 Analysis and Discussion

Per-call logs let us test which observed behaviours distinguish successful runs. None of the five countable metrics does so at the model level. Behavioural metrics come from the same cells as the outcome scores, and conclusions sensitive to task composition are re-checked on the 91-task common subset (Appendix D).

6.1 Tool-use strategy clusters by vendor

Given forty photographs and a shell, a model can look at them or it can write a script. The ratio of image-reading to code-execution calls per task spans **0.58 to 11.36** across 27 models, a factor of 19.6. Gemini-3.8-Flash writes code for almost everything; Muse-Spark-1.2 reads images eleven times more often than it writes code.

After splitting models into terciles by that ratio, we find no significant pass-rate difference between the two ends, while their tool error rates differ 3.0-fold. A high tool error rate is therefore not sufficient evidence of a worse agent in this sample (Appendix D).

If strategy were primarily a response to the task, vendors should interleave. Instead, a one-way ANOVA of log strategy ratio on vendor, over the 25 models from the nine vendors with at least two models, gives η^2 **= 0.825** (F = 9.40, df = 8,16, p = 8.9e-5), and all five Gemini models occupy a band no other vendor enters. This clustering is consistent with vendor-specific training or scaffolding, while the measured choice is not significantly associated with success (Appendix D).

6.2 Countable trajectory metrics show no detectable association with success

Five quantities readable straight off a log (vision-to-code ratio, tool calls, tool error rate, trajectory steps, and output tokens) show no significant model-level association with success ($|r| \leq 0.262$, every $p > 0.18$), and all five null results replicate on the common subset. One quantity produced by a judge *reading* the trajectory explains roughly half the variance (r = +0.677, $p < 0.001$; Figure 3, statistics in Appendix D). Process therefore matters, but the relevant information is absent from these five summaries, and reducing calls or tokens alone should not be expected to improve correctness. Figure 7 in Appendix D gives labelled efficiency views for the same frozen model set.

Pairing outcome and process on the 3,393 cells that have both (process threshold 0.8), **35.4% of all failures occurred in runs with process scores of at least 0.8**. The cell-level correlation is only r = +0.224 against r = +0.677 at the model level. The stronger between-model association does not establish that process score causes success on a task. The score is also partly blind: two Anthropic generations are nearly identical in process yet differ by 0.222 in outcome on the tasks that separate them (Appendix D): per-item accuracy sustained over dozens of items leaves almost no trace in a trajectory.

The process score is also not neutral with respect to strategy, so it should not be used for training or selection without additional calibration for strategy-related bias; Appendix F gives the case. At the model level it remains a useful diagnostic of how agents work; for a single run, only the delivered artifact settles whether the task succeeded.

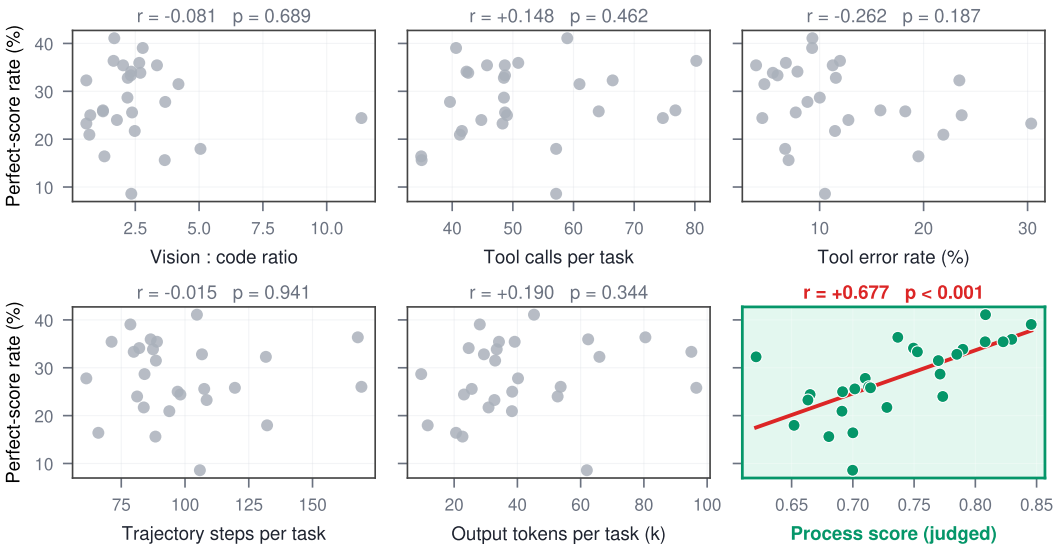

Figure 3: **Model-level associations between trajectory summaries and perfect-score rate.** Each point is one of the 27 evaluated models. The five grey panels plot quantities readable directly off a log against the perfect-score rate; none has a significant correlation. The sixth plots the score a judge assigns after *reading* the trajectory.

6.3 The batch–horizon inversion

Outcome splits sharply by mode. Batch processing (62 tasks, 1,628 cells) reaches a **16.0%** perfect rate at mean 0.629; synthesis (51, 1,369) reaches 36.2% at 0.732; long-horizon interaction (16, 417) reaches **46.0%** at 0.787.

The mode often regarded as especially challenging for current agents (act, observe, correct) is the one models handle best here, at **2.9 times** the perfect-score rate of plain batch work. The orthogonal `work_type` label agrees: the 6 tasks requiring environment manipulation reach 64.9% against 20.6% for the 123 answer-producing tasks.

We offer an explanation and mark it as a hypothesis, since the failure-attribution analysis that would establish it is not yet done. Long-horizon tasks supply a verifiable intermediate state and an unambiguous termination condition, so a model can check itself; batch tasks supply neither, and nothing signals that item twenty broke convention. The exacting tasks of Section 3.4, on which models complete most of the work yet rarely achieve a perfect score, fit this reading: 11 of the 15 are batch tasks, against 62 of 129 in the pack. The receipt ledger of the opening illustrates the pattern. In a 40-receipt task, one model misclassifies 11 receipts yet receives a higher process score (0.85) than a run that gets all 40 right (0.80; Appendix D.4); both trajectories look orderly, and nothing in the environment flagged the eleven errors. Two caveats: 46.0% rests on 16 tasks and carries a wide interval, and long-horizon here means visual puzzles and artifact authoring against a target, not the multi-day autonomous operation other work studies.

6.4 Implications

For benchmark design, the results argue for verifying the delivered artifact item by item: the dominant failure leaves little trace in trajectory summaries (Section 6.2), and the two aggregations of the same outcomes rank models differently (Section 5.1). Difficulty, too, should be measured rather than assigned: author labels agree with the measured bands of Section 3.4 on only 26 of 129 tasks (Appendix D.2), so the pack ships the measured label. Efficiency should be reported separately from capability, because fewer calls or tokens do not track success.

For agent developers, the inversion points to a concrete lever. Batch tasks lack the intermediate checks that interactive tasks supply, so an agent can recover part of that signal by inserting its own, such as recomputing totals, re-reading a sample of items, or checking that every input appears exactly once in the output. Each mirrors a kind of `code` criterion in the rubric, so its effect can be measured directly on this pack. Recommendations are in Appendix G.

6.5 Limitations

Each cell is a single temperature-0 run, and adjudication uses one fixed judge model; cross-judge calibration remains future work. The long-horizon result rests on 16 tasks, and our explanation of the batch–horizon inversion is a hypothesis pending failure attribution. Provider updates and routing were recorded but not experimentally controlled (Appendix F). Two of the main threats to validity are bounded directly: the paired harness control (Appendix B) addresses dependence on the agent loop (pooled shift -0.015), and the 91-task intersection (Section 5.1) addresses dependence on which tasks each model completed, leaving the ordering essentially unchanged ($\tau = 0.864$).

7 Conclusion

DeliverBench evaluates whether multimodal agents can turn batches of authentic material into deliverables that downstream programs can use, verifying each item under a fixed, general tool surface. Across 27 frontier models, the best passes 41.1% of tasks against 87.0% for humans, and failures concentrate in batch work, where one inconsistent item silently invalidates the artifact; 11 of the 15 tasks on which models do most of the work yet rarely finish are batch tasks. Countable efficiency summaries do not track success, and tool-use strategy reflects the vendor more than the task. Because every criterion is tied to an item in the delivered artifact, each failure can be traced to the item that caused it, so the pack supports diagnosis as well as ranking. The 53 tasks no model solves, at least 36 of which people do solve, give the next generation of agents a concrete target. We release the pack, criteria, and logs so that cross-item consistency can become a measured target for agent development. Two directions follow directly: attributing failures to specific items to test the consistency hypothesis, and training or prompting agents to verify their own intermediate outputs, such as running totals and item counts, when the environment provides no such signal.

AI USE STATEMENT

In this work, we used generative AI tools as the subject of study: all 27 evaluated systems are language models, and one language model serves as the rubric judge, as described in Section 4. We additionally used AI assistance for drafting and editing prose and for writing analysis and figure-generation scripts, all of which were reviewed by the authors. Every number reported in this paper is recomputed from the frozen data snapshot by the released scripts rather than produced by a model. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The task pack, rubrics, and the frozen data snapshot are provided in the supplementary material; the logged trajectories will be released upon acceptance. Every table and figure in this paper is regenerated from that snapshot by the released scripts. Run configuration and the tool inventory are given in Appendix C, the harness-invariance control in Appendix B, and the human-baseline protocol, fixed before collection, in Appendix E.

REFERENCES

Rahul K. Arora et al. HealthBench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, Manasi Sharma, Andrew Park, Martin Dimakis, Ernesto Gabriel Hernandez Montoya, Dan Rambado, Ivan Salazar, Rafael Cruz, MohammadHossein Rezaei, Chetan Rane, Ben Levin, Daniel Yue Zhang, Brad Kenstler, and Bing Liu. MCP-Atlas: A large-scale benchmark for tool-use competency with real MCP servers. *arXiv preprint arXiv:2602.00933*, 2026.

Victor Barres et al. τ^2-Bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Yuchen Mao, Wenjing Hu, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. Spider2-V: How far are multimodal agents from automating data science and engineering workflows?, 2024. URL https://arxiv.org/abs/2407.10956.

CocoaBench Team, Shibo Hao, Zhining Zhang, Zhiqi Liang, Tianyang Liu, Yuheng Zha, Qiyue Gao, Jixuan Chen, Zilong Wang, Zhoujun Cheng, Haoxiang Zhang, Junli Wang, Hexi Jin, Boyuan Zheng, Kun Zhou, Yu Wang, Feng Yao, Licheng Liu, Yijiang Li, Zhifei Li, Zhengtao Han, Pracha Promthaw, Tommaso Cerruti, Xiaohan Fu, Ziqiao Ma, Jingbo Shang, Lianhui Qin, Julian McAuley, Eric P. Xing, Zhengzhong Liu, Rupesh Kumar Srivastava, and Zhiting Hu. CocoaBench: Evaluating unified digital agents in the wild, 2026. URL https://arxiv.org/abs/2604.11201.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, 2023.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. SWE-bench Pro: Can AI agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Ling Fu et al. OCRBench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8c2e6bb15be1894b8fb4e0f9bcad1739-Abstract-Datasets_and_Benchmarks_Track.html.

Xingang Guo, Utkarsh Tyagi, Advait Gosai, Paula Vergara, Jayeon Park, Ernesto Gabriel Hernández Montoya, Chen Bo Calvin Zhang, Bin Hu, Yunzhong He, Bing Liu, et al. Beyond seeing: Evaluating multimodal LLMs on tool-enabled image perception, transformation, and reasoning. *arXiv preprint arXiv:2510.12712*, 2025. VisualToolBench.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018. URL https://arxiv.org/abs/1803.02324.

Shibo Hao, Zhining Zhang, Zhiqi Liang, Tianyang Liu, Zilong Wang, et al. CocoaBench: An evaluation framework for general agents with compositional cognitive abilities. Blog post, December 2025. URL https://cocoabench.github.io/.

Mingxin Huang, Yongxin Shi, Dezhi Peng, Songxuan Lai, Zecheng Xie, and Lianwen Jin. OCR-Reasoning benchmark: Unveiling the true capabilities of MLLMs in complex text-rich image reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aH7eyx64pC.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. FigureQA: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*, 2017.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. SheetCopilot: Bringing software productivity to the next level through large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.19308.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=z53s5p0qhf.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024b. doi: 10.1007/s11432-024-4235-6.

Zhiqiang Liu, Wenhui Dong, Yilang Tan, Yuwen Qu, Haochen Yin, and Chenyang Si. TOBench: A task-oriented omni-modal benchmark for real-world tool-using agents, 2026. URL https://arxiv.org/abs/2605.16909.

Zixian Ma, Weikai Huang, Jieyu Zhang, Tanmay Gupta, and Ranjay Krishna. m&m's: A benchmark to evaluate tool-use for multi-step multi-modal tasks, 2024. URL https://arxiv.org/abs/2403.11085.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022. URL https://arxiv.org/abs/2203.10244.

Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021. URL https://arxiv.org/abs/2007.00398.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and C.V. Jawahar. InfographicVQA. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022. URL https://arxiv.org/abs/2104.12756.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25ae35b5b1738d80f1f03a8713e405ec-Abstract-Conference.html.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. OCR-VQA: Visual question answering by reading text in images. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2019.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. GDPval: Evaluating AI model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025.

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

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 56843–56873. PMLR, 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

Zhaochen Su, Jincheng Gao, Hangyu Guo, Zhenhua Liu, Lueyang Zhang, Xinyu Geng, Shijue Huang, Peng Xia, Guanyu Jiang, Cheng Wang, Yue Zhang, Yi R. Fung, and Junxian He. AgentVista: Evaluating multimodal agents in ultra-challenging realistic visual scenarios, 2026. URL https://arxiv.org/abs/2602.23166.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://arxiv.org/abs/2407.18901.

Jize Wang, Zerun Ma, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. GTA: A benchmark for general tool agents, 2024a. URL https://arxiv.org/abs/2407.08713.

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.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. OfficeBench: Benchmarking language agents across multiple applications for office automation, 2024b. URL https://arxiv.org/abs/2407.19056.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=tN61DTr4Ed.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://arxiv.org/abs/2311.16502.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. IFEval.

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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

Yang Zhou, Mingyu Zhao, Zhenting Wang, Difei Gu, Bangwei Guo, Ruosong Ye, Ligong Han, Can Jin, and Dimitris N. Metaxas. M^3-Bench: Multi-modal, multi-hop, multi-threaded tool-using MLLM agent benchmark, 2025. URL https://arxiv.org/abs/2511.17729.

Xuanyu Zhu, Yuhao Dong, Rundong Wang, Yang Shi, Zhipeng Wu, Yinlun Peng, YiFan Zhang, Yihang Lou, Yuanxing Zhang, Ziwei Liu, Yan Bai, and Yuan Zhou. VTC-Bench: Evaluating agentic multimodal models via compositional visual tool chaining, 2026. URL https://arxiv.org/abs/2603.15030.

A Shortcut Screening of the Released Pack

A benchmark score is evidence about a capability only if the tasks cannot be solved without it (Raji et al., 2021). The governing construction principle is therefore **visual grounding**: every value a

model is asked to produce must be obtainable only by reading the source material. To enforce it, the pack was screened before release by six complementary tests. Each of the four cases below was caught by exactly one test and would have survived the other five.

1. **Image-removal test.** Delete the images; check whether the answer is still derivable from text alone. This is the multimodal analogue of the hypothesis-only baseline that exposed annotation artifacts in natural language inference (Gururangan et al., 2018).
2. **Key-and-value-on-the-same-row check.** The most reliable single indicator of leakage.
3. **Assets-as-answer test.** Feed each non-image file to the judge *as if* it were the model's submission, and see what it scores.
4. **Vector and text-format asset scanning.** SVG and HTML are text and need three passes: literal matching of text nodes; testing whether the gold value is a **sum of several numbers** present in the file; and inspection of comments and geometry attributes.
5. **Text-only solvability.** Hand-write a script that reads only the non-image files and compare its output field by field against the gold answer.
6. **Field-level manual review.** For screenshot tasks, look at every image and ask whether the answer field is already displayed.

Each test catches something the others miss. Test 5 is the only one that detects a task whose every gold field is reproducible by a script that never opens an image. Visual grounding can fail completely while each individual field still looks image-derived. Test 1 is the only one that exposes a *distributional* shortcut, where a rule such as "a price is present, therefore approve" happens to cover every row; a task can survive literal-value scanning and still be answerable without looking. Test 3 catches auxiliary files that duplicate the gold answer in another form, such as timestamps that match it line for line. Test 4's second pass catches the case a literal scan cannot see at all: an SVG containing 42, 8 and 2 against a gold answer of 52, where **no single value matches and only their sum does**.

A leakage scan that looks for the answer as a literal string would report that task as clean.

B Harness Sensitivity Control

Every agent benchmark reports scores produced by some agent loop, and every such benchmark is therefore open to the objection that it measures the loop rather than the model. Scaffold effects can be large (CocoaBench Team et al., 2026), but their size is specific to each benchmark. Holding the task set, the scoring rubric, the judge model, and the API gateway fixed, we re-ran a subset of models under two vendor-built agent harnesses: Anthropic's Claude Code CLI and OpenAI's Codex CLI. Each uses its native tool surface and is proxied to the same gateway. We compared each against our own loop on the same cells. Only cells valid on both sides of a pair are compared; 16 visual-game pairs are excluded because our loop exposes a game-state hook the vendor CLIs have no equivalent for, and including them would compare instrumentation rather than models.

Table 3: Paired comparison of the same models run under three harnesses on common tasks.

Harness	Model	Paired n	Harness mean	Ours	Δ
Claude Code	Claude Opus 4.8	94	0.775	0.787	-0.011
Codex	GPT-5.5-xHigh	81	0.746	0.752	-0.006
Codex	GPT-5.6-Sol-Max	63	0.788	0.797	-0.009
Codex	GPT-6-Astra-Max	80	0.786	0.818	-0.033

Pooled over 318 paired cells the difference is $\Delta = -0.015$, 95% CI $[-0.038, +0.009]$, which is not significant. Perfect-score outcomes comprise 221 ties, 60 losses, and 37 wins. Mean rubric score is lower under the vendor harness in 91 pairs, and the worst decile accounts for 77.5% of the total shortfall. Three tasks recur across harnesses (`feedback-rule-config-validation` -0.45, `web-hero-layout-correction` -0.32, `customer-call-intent-grouping` -0.32); all three are artifact-authoring tasks where our sandbox supplies a preview channel the CLIs do not have, which is a difference in environment rather than in model.

B.1 Score comparison across tool surfaces

With every harness instrumented, the vendor CLIs' trajectories are recorded on the same footing as ours, allowing the score comparison to be interpreted alongside tool-use differences.

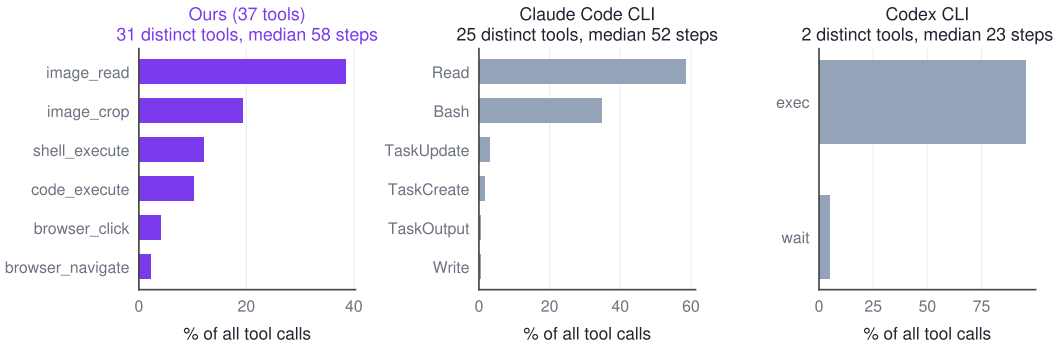

Figure 4: **Tool-use distributions under three agent harnesses.** Bars show each harness's six most-used tools as a share of all its tool calls.

Table 4: Tool surfaces differ sharply across harnesses while scores do not.

Harness	Cells	Median steps	Median tool calls	Distinct tools used
Ours	472	58.0	35	31
Claude Code CLI	112	52.5	29	25
Codex CLI	279	23	11	**2**

Codex does essentially everything through `exec` (3,867 of 4,061 calls, with `wait` accounting for the rest). It has **no dedicated image-reading tool**; image access, when needed, is mediated through `exec`. Our loop spends 38% of its calls on `image_read` and 19% on `image_crop`; Claude Code concentrates on `Read` (58%) and `Bash` (35%). The pooled score difference is not significant despite these tool-surface differences, although the confidence interval and concentrated task-level losses rule out a general equivalence claim.

An instrumentation caveat. Both vendor CLIs report a tool error rate of exactly zero, against 2.6% for our loop. We do not read this as flawless execution; their transcripts do not surface failed calls the way ours does. Error rates are therefore comparable *within* our harness, as used in Section 6.1, but not *across* harnesses.

This control does not establish that harnesses are equivalent. It shows that the mean difference is small and not significant for these four model–harness comparisons, while also identifying artifact-authoring tasks on which harness choice matters.

C Tool Surface and Run Configuration

The 37 tools. None is task-specific; a task about invoices and a task about warehouse photographs see exactly the same set.

Asset access. Task assets are mounted into the sandbox and accessed through tools rather than injected directly into the prompt. This is the operational setting we use for multimodal *tool-call* evaluation, and it makes tool use auditable: a model that never looked at an image cannot conceal that it did not.

Run configuration. All main experiments use one profile, applied identically to every model.

Behavioural columns. The behavioural columns of the leaderboards in Section 5 and Appendix D.13 are per-task means taken from the trajectory logs over the same cells that carry the scores: total tool calls, output tokens in thousands, and calls broken out by family (image read and crop, code and shell execution, browser and DOM operations, and file reads and writes).

Table 5: The tool inventory. Every model sees the same set on every task.

Family	n	Tools
Browser	15	`browser_click`, `browser_type`, `browser_press`, `browser_scroll`, `browser_move_to`, `browser_move_rel`, `browser_drag_to`, `browser_drag_rel`, `browser_hotkey`, `browser_key_down`, `browser_key_up`, `browser_wait`, `browser_screenshot`, `browser_get_viewport_info`, `browser_navigate`
DOM	10	`dom_get_text`, `dom_get_html`, `dom_query_selector`, `dom_extract_links`, `dom_mark_elements`, `dom_click`, `dom_hover`, `dom_type`, `dom_press`, `dom_scroll`
File	7	`file_read`, `file_write`, `file_list`, `replace_in_file`, `search_in_file`, `find_files`, `str_replace_editor`
Image	2	`image_read`, `image_crop`
Execution	2	`code_execute`, `shell_execute`
Termination	1	`task_complete`

Table 6: Run configuration, applied identically to every model.

Setting	Value	Rationale
Max iterations	200	Backstop, not intended to bind normal completions
Max total tokens	unlimited	Only the iteration cap binds
Temperature	0.0	Providers with adaptive thinking may ignore this
Image eviction from context	off	No image is dropped once read
Max context images / MB	200 / 512	Ceiling only; inactive while eviction is off
Auto debug screenshot	captured, not sent to model	For reports only

Prompt neutralisation. The neutral prompt states the task, directs the model to inspect images with `image_read` and to use `image_crop` for fine detail, notes that inputs remain on disk and can be re-read, forbids inventing values, and specifies the termination contract. One provider (Qwen3-VL) retains its own template because it requires tool descriptions inlined as text.

Sandbox. Each task runs in its own Docker container built from the task's own compose file, based on a common sandbox image. The sandbox HTTP API is bound to host loopback only. Containers have ordinary outbound network access: there is no dedicated search service, and models reach the public web through the browser tools or through `wget`/`curl` in the shell. Attachments are copied into `/workspace/inputs` after the container reports healthy, and ownership is adjusted for the unprivileged sandbox user.

Per-model image budgets. A small number of models carry per-model overrides on the context image budget, necessitated by gateway request-size limits rather than by experimental design. For providers where inlined base64 would overflow the gateway, images are uploaded once to object storage and passed by URL, which keeps the request body small while preserving what the model sees. These overrides are recorded per runset in the released snapshot.

D Additional Analyses

D.1 Resolution: capability jumps versus incremental releases

We run paired t-tests on the common task subset for each adjacent release pair:

The resolution of this benchmark lies between roughly 7 and 14 points of pass rate. One generational step is detected cleanly (+14.3 points); the next largest, +7.1 points, is not, and nothing below it is. We report the bracket rather than a single number because the data does not support a sharper claim, and because a leaderboard without a stated resolution invites over-reading. Failing to detect a difference is not evidence that two models are equivalent.

Table 7: Paired t-tests on common task subsets for every adjacent pair of releases.

Comparison	n	Δ pass rate	p	Δ mean	p (mean)
Kimi-K3 − Kimi-K2.6	126	**+14.3 pt**	**0.0003**	+0.176	**<0.001**
Gemini-3.8 − 3.7	127	+7.1 pt	0.072	+0.028	0.331
Claude Opus 5 Max − Opus 4.8	128	+5.5 pt	0.090	+0.026	0.170
Doubao-2.1 − 2.0	128	+5.5 pt	0.090	+0.044	0.145
Grok 4.6 − 4.5	127	+2.4 pt	0.534	−0.057	**0.023**
Gemini-3.7 − 3.6	128	+1.6 pt	0.639	+0.015	0.497
Gemini-3.6 − 3.5	129	+1.6 pt	0.619	+0.054	0.057
GPT-5.6 − GPT-5.5	129	−0.8 pt	0.820	+0.014	0.536

Two further details stand out. GPT-5.5→5.6 moves the pass rate *backwards* by 0.8 points across 129 shared tasks while the mean improves by 0.014, providing no evidence of a pass-rate improvement on this task family while remaining compatible with small effects in either direction. And Grok 4.5→4.6 splits by metric in the opposite direction from the usual: the mean score *drops* significantly ($p = 0.023$) while the pass rate rises 2.4 points, which is the phenomenon of D.5 appearing in a version comparison.

D.2 Author-assigned difficulty versus measured difficulty

Every task also carries a difficulty assigned by its author, which the evaluation finds carries almost no information about completion.

Table 8: Author-assigned difficulty labels against measured outcome.

Labelled difficulty	Tasks	Perfect rate	Mean
easy	18	25.9%	0.752
medium	54	**31.4%**	0.709
hard	57	23.0%	0.644

The ordering is not even monotonic: tasks the authors called *medium* have the highest perfect-score rate, five points above *easy*. Mean score does decline monotonically, so the label weakly predicts how much partial credit a task yields, but not whether it can be completed, which is the quantity of interest. Against the measured difficulty bands of Section 3.4, author labels and measured bands agree on 26 of 129 tasks (20%). This is why the pack ships the measured label and why anyone stratifying a sample on hand-assigned difficulty should validate it first.

D.3 Process metrics versus the pass rate

The correlations plotted in Figure 3, with the test statistics. Each row is over the 27 evaluated models.

Table 9: Countable properties of a trajectory against the perfect-score rate, across the 27 models. Only the judged process score, which requires reading the trajectory, predicts the outcome.

Process metric	r with pass rate	t	p
Vision : code ratio	−0.081	−0.40	0.689
Tool calls per task	+0.148	+0.75	0.462
Tool error rate	−0.262	−1.36	0.187
Trajectory steps per task	−0.015	−0.07	0.941
Output tokens per task	+0.190	+0.97	0.344
Process score (judged)	**+0.677**	**+4.60**	**<0.001**

D.4 Limits of the process score

The two Anthropic generations provide a concrete example for Section 6.2 because they are close in aggregate outcome and nearly identical in process. Across the 128 tasks both attempted, Claude

Opus 5 Max alone succeeds on 12 and Claude Opus 4.8 alone on 5. On the 12 decisive tasks the two process scores are 0.853 and 0.855 (a 0.002 difference), while the outcome scores are 0.778 and 1.000.

Table 10: Three tasks that separate two Anthropic generations on outcome while the process score stays flat or inverts.

Task	Opus 4.8 outcome / process	Opus 5 Max outcome / process
`table-screenshot-to-json`	0.899 / 0.84	1.000 / 0.84
`travel-itinerary-to-schedule`	0.940 / 0.84	1.000 / 0.85
`expense-receipt-screening`	0.786 / **0.85**	1.000 / **0.80**

The first task asks for 90 records of 11 fields; Opus 4.8 returns 100 incorrect field values and Opus 5 Max none, and the judge gives both 0.84. In the third the ordering inverts: Opus 4.8 misclassifies 11 of 40 receipts and scores 0.85 on process, above the 0.80 given to the run that got every receipt right. The measured failure, per-item accuracy sustained over dozens of items, leaves almost no trace in the trajectory, because the trajectory of a model that transcribes 990 fields correctly looks like that of one that transcribes 890.

D.5 Aggregation choice and model ranking

The two aggregations rank models differently, and the gaps run both ways: Gemini-3.6-Flash is 13th by mean rubric score but 20th by pass rate (reliably close, rarely exact), while Qwen3.8-Flash-xHigh inverts it, 22nd by mean and 15th by pass rate. Paired tests against DeepSeek-V4-Flash-Vision show the same split inside single comparisons: Claude Fable 5 leads by +0.032 mean (ns) but +10.7 points of pass rate ($p = 0.004$), and Kimi-K3 by +0.055 mean (ns) but +8.2 points ($p = 0.025$). For two of three pairs, one aggregation is significant while the other is not, so which aggregation a benchmark reports affects which model appears better supported.

D.6 Strategy terciles on the common task subset

The strategy analysis of Section 6.1 is re-run on the 91 tasks all 27 models completed, to confirm the pattern is not an artifact of differing task composition. Ratios shift upward on this subset (0.67–18.58 rather than 0.58–11.36) because the excluded tasks are disproportionately code-friendly, but the pattern is unchanged: pass rates of 26.1% / 32.2% / 25.3% across the three terciles, with the two ends not significantly different, and the tool-error gradient intact (17.7% / 7.8% / 6.0%). The log-ratio–to–error-rate correlation is $r = -0.905$ on this subset against $r = -0.889$ on the full set.

D.7 Strategy terciles on the full pack

Table 11: Models grouped into terciles by vision-to-code ratio. Approach varies enormously; outcome does not follow it.

Tercile	n	Ratio range	Pass rate	Tool error rate
Code-leaning	9	0.58–1.68	27.5%	**19.4%**
Middle	9	1.78–2.48	27.1%	9.9%
Vision-leaning	9	2.66–11.36	29.1%	**6.3%**

D.8 Vendor clustering of tool-use strategy

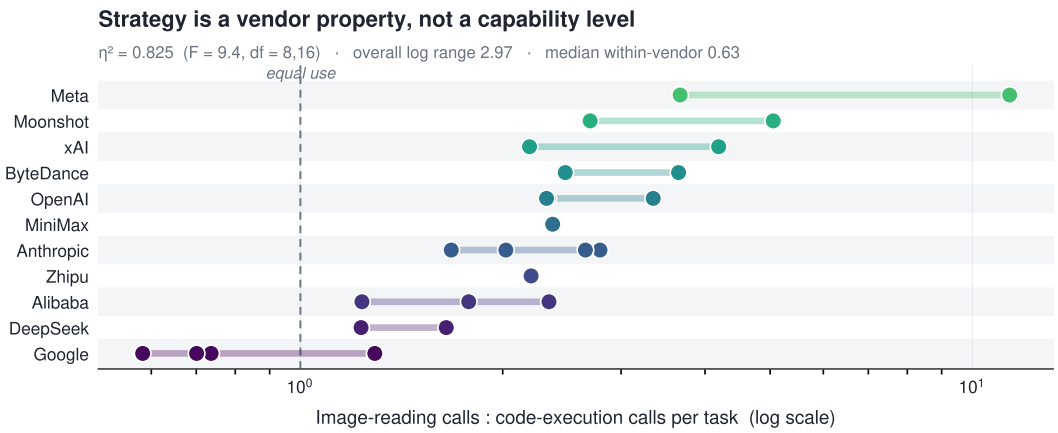

Figure 5: **Tool-use strategy clusters by vendor.** Each point is a model, positioned by its ratio of image-reading to code-execution calls per task (log scale) and grouped by vendor. Vendors cluster tightly even across large capability differences; the ratio is not significantly associated with pass rate.

D.9 Outcome grid

Figure 2 shows the full 27 × 129 outcome grid. Almost every cell in its 53-task unsolved band is scored-but-imperfect rather than empty, indicating a completed but non-perfect submission.

D.10 Worked task example

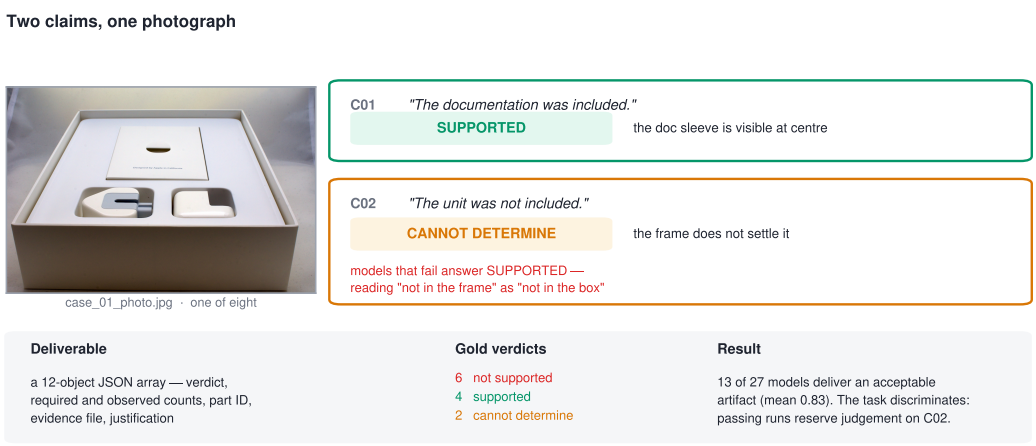

Figure 6: **Worked example of item-level visual adjudication.** Eight unboxing photographs yield twelve claims, each adjudicated against its own photograph. C01 and C02 concern the same photograph but have different answers: a part absent from the frame is not thereby missing. In the illustrated failure mode, the second verdict is collapsed into the first.

D.11 Released pack summary

Table 12: The released pack at a glance.

Dimension	Value
Tasks	129
Source assets	2,339 files / 2.96 GB (1,840 PNG, 326 JPG, 39 CSV, 22 WAV, 21 MP4, 20 MP3, 16 PDF)
Assets per task	median 10, mean 18.1, max 240
Asset access	mounted in the sandbox and accessed through tools
Rubric criteria	411 (median 3 per task, range 1–8)
Modes	batch 62 / synthesis 51 / long-horizon 16
Models evaluated	27
Valid outcomes	3,414 of 3,483 reachable (98.0%)

D.12 Data-path consistency

Scores are read directly from the 3,414 run databases rather than from an ingested warehouse copy. The warehouse lags live execution while re-adjudication writes back to the run database, and the two diverge: of the 2,216 cells present in both, **225 disagree (10.2%)**. We take the run database as authoritative, and the released statistics are regenerated from it rather than from the warehouse. An evaluation report assembled from more than one data path should state which path it reports.

D.13 Full ranked leaderboard

Table 13: Twenty-seven models and a human baseline on the same 129 tasks. *Mean* is the mean rubric score and *Perfect %* the share of tasks scoring ≥ 0.999, which is also the pass rate. The remaining columns are per-task means over the same cells, taken from *Proc* is the independent process score, which reads the trajectory and never sees whether the task succeeded. The rest are per-task means over the same cells, taken from the logged trajectories: total tool calls, output tokens in thousands, and calls by tool family. Human sessions were not instrumented, hence the empty columns.

#	Model	n	Mean	Perfect %	Proc	Calls	kTok	Image	Code	UI	File
—	**Human**	129	**0.953**	**87.0**	—	—	—	—	—	—	—
1	Claude Opus 5 Max	129	0.804	**41.1**	0.808	59	45	31	19	5	2
2	DeepSeek-V4.1-Flash(max)	121	0.774	36.4	0.737	80	80	43	26	7	3
3	Claude Opus 4.8	128	0.784	35.9	0.830	51	62	31	11	5	3
4	GPT-6-Astra-Max	127	0.784	35.4	0.823	46	34	29	9	5	2
5	Claude Fable 5	127	0.742	35.4	0.808	49	39	27	13	6	2
6	GPT-5.5-xHigh	129	0.735	34.1	0.750	42	25	23	10	6	3
7	Kimi-K3-Thinking	127	0.760	33.9	0.790	43	33	27	10	3	2
8	GPT-5.6-Sol-Max	129	0.748	33.3	0.753	49	95	26	11	7	3
9	GLM-5.3-Flash-Max	128	0.755	32.8	0.785	49	29	29	13	3	2
10	Gemini-3.8-Flash	127	0.761	32.3	0.621	66	66	22	38	3	2
11	Grok 4.6 xHigh	127	0.700	31.5	0.770	61	33	42	10	6	3
12	Grok 4.5 High	129	0.756	28.7	0.771	49	10	30	14	4	0
13	Muse-Spark-1.1-xHigh	126	0.697	27.8	0.710	40	40	26	7	3	2
14	DeepSeek-V4-Flash-Vision	123	0.708	26.0	0.713	77	54	35	29	9	3
15	Qwen3.8-Flash-xHigh	124	0.618	25.8	0.714	64	97	32	26	4	1
16	MiniMax-M3	129	0.637	25.6	0.702	49	26	29	12	6	1
17	Gemini-3.7-Flash	128	0.732	25.0	0.692	49	38	18	25	4	1
18	Muse-Spark-1.2-xHigh	127	0.682	24.4	0.665	75	23	65	6	2	2
19	Qwen3.8-Max-0902	125	0.704	24.0	0.773	45	53	25	14	4	2
20	Gemini-3.6-Flash	129	0.721	23.3	0.663	48	33	15	26	5	2
21	Doubao-Seed-2.1-Pro	129	0.660	21.7	0.728	42	31	23	9	7	2
22	Gemini-3.5-Flash	129	0.664	20.9	0.691	41	38	14	20	4	3
23	Kimi-K2.6	128	0.578	18.0	0.652	57	12	39	8	9	1
24	Gemini-3.1-Pro	128	0.549	16.4	0.700	35	21	17	13	4	1
25	Doubao-Seed-2.0-Pro	128	0.617	15.6	0.680	35	23	18	5	10	2
26	Qwen3.8-Max-Thinking	128	0.281	8.6	0.700	57	62	35	15	3	3

`Calls`, `kTok`, and the four tool-family columns are per-task means over each model's valid cells. Human sessions were not instrumented. Claude Fable 5.1 Max is excluded because only 105 tasks were valid; Appendix F reports its bounded comparison.

D.14 Efficiency views

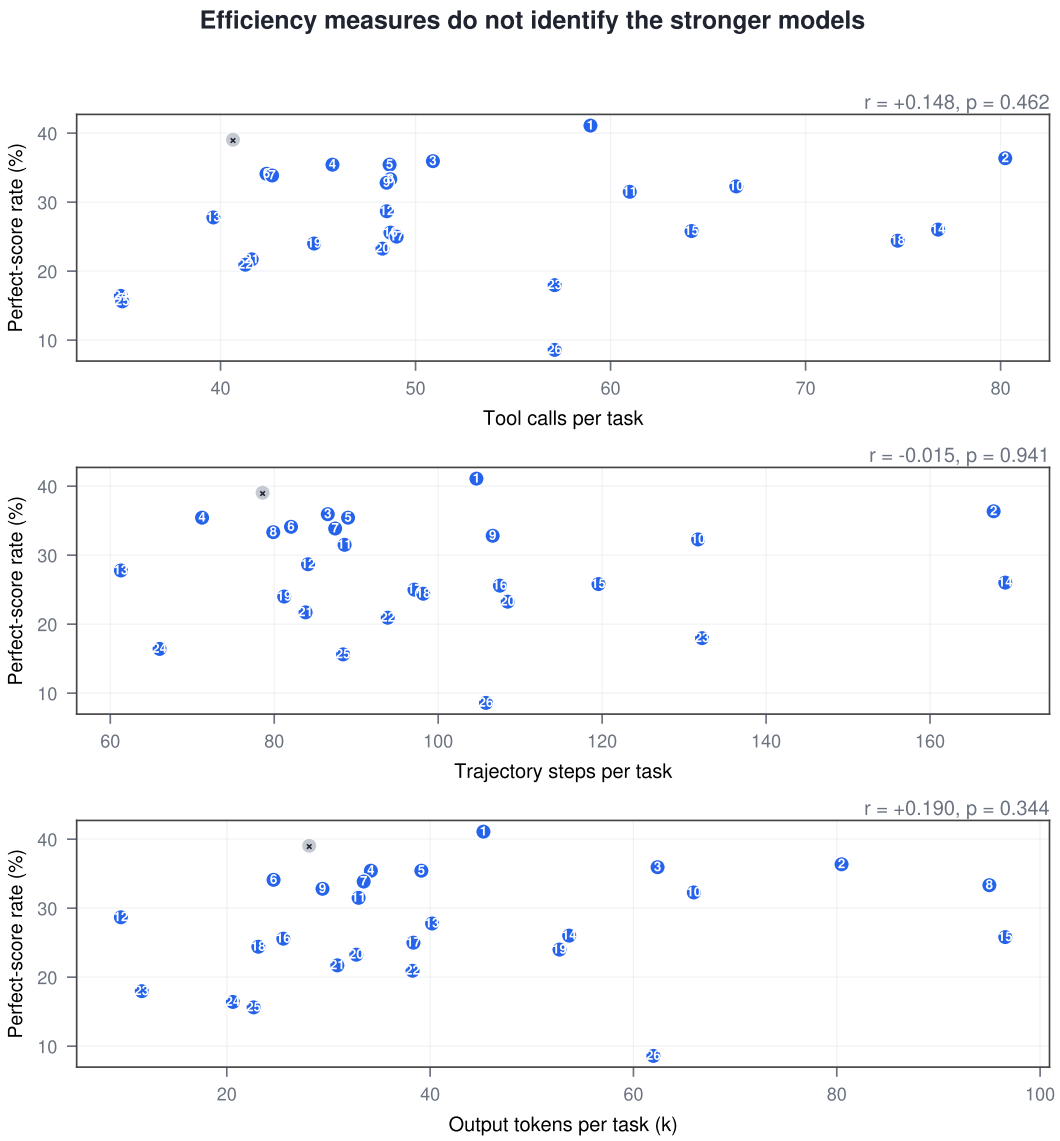

Figure 7: **Model-level resource measures versus perfect-score rate.** Each numbered point uses the rank in Table D.13; the grey × is the low-coverage Claude Fable 5.1 Max. All three panels use the frozen 27-model paper scope. Calls, trajectory length, and output tokens have no significant model-level correlation with the perfect-score rate. We omit total-token and reasoning-token panels because provider accounting is not comparable for the historical runs.

E Human Baseline Protocol

The protocol below was fixed before collection began. It is recorded in full so that the result reported in Section 5 cannot be chosen after the fact. Collection in the end covered all 129 tasks, which makes

the sampling and weighting machinery below moot for the reported number. It is retained because the prespecified readings are what license the interpretation in Section 5.

Planned sample and weighting. The study was designed around a 45-task stratified sample drawn on measured perfect-score rate, with the hardest stratum deliberately oversampled. Weights were to be computed on the joint (perfect-rate stratum $\times$ task mode) cell rather than on the stratum alone, because the two oversampling decisions push in opposite directions (harder on difficulty, easier on mode), and stratum-only weights do not correct both. On a known quantity the weighted estimator puts Claude Opus 5 Max at 39.8% against its true 41.1%, while reading the same 45 tasks unweighted gives 33.3%, an 8-point error in the direction the oversampling predicts. Full coverage removed the need for any of this: the human row in Section 5 is a direct measurement on the same 129 tasks as every model row.

Staged collection. The sample was nested: pilot (6 tasks, 4 of them solved by no model) $\subset$ reduced (24) $\subset$ full (45) $\subset$ complete pack (129). The pilot was read against a rule fixed in advance:

- 5 or 6 of 6 solved $\rightarrow$ proceed to the reduced set.
- 4 of 6 $\rightarrow$ expand to 12 tasks before deciding.
- 3 or fewer $\rightarrow$ the planning assumption was wrong; re-evaluate the design rather than collect more.

Staging guarded against a wrong planning assumption, which a larger sample cannot correct. Had humans scored near 50%, even 45 tasks would have given an interval overlapping the model rates, and the study would have consumed its budget without being able to answer its question. The observed rate was far enough above that collection was carried through to the whole pack.

Analysis plan. Five outcomes were enumerated in advance together with their readings, so that no result could be reinterpreted favourably after it arrived.

Table 14: Readings of the human-baseline result, fixed before collection began.

Outcome	Reading
Humans far above the best model	Evidence of a capability gap under this protocol, subject to the judge and annotator limitations in Appendix F
Humans near the best model	The bar is strict rather than the models weak; the rubrics need re-examination before the gap claim can stand
Humans below the best model	Evidence that the rubric or task specification requires re-examination, not evidence of a model deficit
Humans high on batch tasks, not on long-horizon	Section 6.3 describes a model deficit
Humans low on batch tasks too	Section 6.3 describes the work itself, not the models

The observed result, 87.0% against 41.1%, is the first row. The two mode-conditional rows remain open pending the per-mode breakdown.

Conditions. Annotators receive the same prompt text and the same source files the models receive, and work directly rather than through the agent loop; they are given a per-task time budget derived from the median model trajectory length. They may not consult the gold answer or any model output. Submissions enter the identical adjudication pipeline with no annotation of their origin.

F LIMITATIONS

The human baseline calibrates the strict pass bar under the same scoring protocol, although it comes from one internal collection without a reported inter-annotator agreement estimate. Adjudication uses one fixed judge model; cross-judge calibration remains future work.

The long-horizon result covers 16 tasks, so conclusions by task mode should be read at that granularity. Measured difficulty labels describe this released pack and support evaluation of future models, rather than inference about the models used to construct the labels.

Claude Fable 5.1 Max is reported but not ranked because its 24 missing cells are concentrated among harder tasks. On the 91 tasks every model completed, it reaches 41.8% perfect (mean 0.819), between Claude Opus 5 Max at 47.3% (0.846) and Claude Opus 4.8 at 38.5% (0.818).

The process score is not neutral with respect to strategy. Gemini-3.8-Flash, the most code-leaning model, has the lowest process score (0.621) yet ranks tenth of 26 on pass rate (32.3%): code-heavy trajectories contain repeated errors and debugging, which the process rubric penalises even when the approach succeeds. Judged process quality should therefore not be used as a training or selection signal without calibrating for strategy.

Finally, each cell is a single temperature-0 run, and provider updates, routing, and image-budget overrides were recorded but not experimentally controlled.

G Appendix G Recommendations Informed by the Controls

Our controls suggest four reporting practices. First, state the aggregation rule and estimate the benchmark's empirical resolution; for our data, adjacent-release comparisons place it between roughly 7 and 14 pass-rate points. Second, keep process and outcome scores separate when analyzing their relationship. Third, match each criterion to a judge that can observe the relevant evidence: text extraction cannot expose workbook metadata or preserved formulas, while visual criteria require a rendering of the model's artifact. Finally, compare independently assembled reporting paths. In our case, this check exposed disagreement in 10.2% of cells before reconciliation.

H Appendix H Criterion-to-Judge Assignment

Judge assignment determines what a criterion can detect. A spreadsheet formula is the clearest case. If the task requires `=SUM(J2:P2)` to be preserved and the criterion is adjudicated from extracted cell *values*, the judge sees an empty cell for exactly the models that complied, because a formula written programmatically carries no cached result. The same applies to sheet protection, data validation and cell locking. All are workbook metadata, none of which survives text extraction. Criteria of this kind get a code judge that opens the workbook and asserts on the formula.

The converse constraint applies to vision criteria. Where a criterion decomposes a rendering into weighted sub-dimensions, each is scored all-or-nothing rather than by continuous deduction: under a pass bar set at a perfect total, a judge that withholds a fraction of a point on aesthetic grounds makes the task unpassable regardless of the artifact. Vision criteria also render the model's own deliverable, including vector and HTML artifacts rasterised for the judge, rather than any input asset the task required the model to package alongside it.