ACTUAL INTELLIGENCE LABS • TECHNICAL REPORT

Research • Report • July 2026

We Ran 154 Controlled Trials of Claude vs. GPT.

Both Scored Perfect.

A blocked, full-factorial designed experiment on frontier coding models,
with a confidence interval on every pass rate and no winner invented that the data did not earn.

Drake Griffith

CTO, Actual Intelligence Labs

actualintelligencelabs.ai/research/claude-vs-gpt-154-run-experiment

---

Abstract. Most model comparisons are one person running one prompt through two tools and reporting which output they liked better. That is not a measurement; it is an anecdote wearing a lab coat. We did the opposite. We ran 154 coding trials as a blocked full-factorial designed experiment pitting Claude Fable 5 against GPT-5.6, treated each run as a Bernoulli trial because model inference is non-deterministic, and put a confidence interval on every pass rate. Both models passed everything solvable: 152 of 152 clean passes on every configuration where a model worked alone, and zero discordant pairs in the paired head-to-head. The result was therefore not a ranking but a *ceiling*. The honest finding is that the exam was too easy, and the next experiment raises the difficulty until failures appear. This report documents the design, the statistics, and—most importantly—what the experiment cannot tell you.

---

1 Why Most Claude-versus-GPT Comparisons Tell You Nothing

The problem is non-determinism. Ask the same model the same coding question twice and you can get two different answers, two different token counts, two different outcomes. A single run tells you what happened once. It tells you nothing about what happens on average, which is the only thing you actually care about when deciding which tool your team ships with. Run the comparison again tomorrow and the winner can flip.

There is a second, quieter problem. The public benchmarks that do use repeated, structured evaluation are running out of headroom. Frontier models now cluster near the top of the standard suites, which makes those suites worse at telling models apart. Contamination compounds it: when benchmark problems, or text closely derived from them, land in a model's training data, the model can recall an answer instead of reasoning to it. The team behind LiveCodeBench [1] documented exactly this for coding, showing that older suites are no longer sufficient on their own and building a time-segmented benchmark of fresh contest problems specifically to dodge the leakage. The same principle drives our own position on evaluation: public benchmarks measure the model, not your system.

2 How to Design a Fair Test of Two AI Coding Models

The two contestants were **Claude Fable 5**, driven through Anthropic's `claude` CLI, and **GPT-5.6**, driven through OpenAI's `codex` CLI. The design is a full-factorial designed experiment, *blocked by task*. The factors are:

- **Model** — Claude Fable 5, GPT-5.6.
- **Reasoning effort** — two to three levels per model (low / medium / high).
- **Harness** — a bare CLI versus a shared agent-instruction harness.

Blocking means every configuration sees every task, so task difficulty cancels out of any paired comparison. Replication is $n = 3$ repetitions per cell on the smaller tasks and $n = 2$ on the greenfield build, because a single run is an anecdote and repeated runs let us treat each outcome as a Bernoulli trial and estimate a real pass rate. Run order is seed-interleaved (seed 1337) so that time-of-day and API-load drift cannot systematically favor one model.

2.1 The tasks

There are nine tasks, and every one was proven solvable before any model touched it.

- **Four seeded-bug tasks** with deliberately subtle defects: an off-by-one in pagination, a naive-versus-timezone-aware datetime bug, a cache key that silently drops a dimension, and an async dedup race.
- **Four feature tickets** written in a strict six-section format.
- **One greenfield CLI build** — a roughly 200-line cost-splitting tool with a reference implementation.

Each task ships a self-test that fails on the broken starting code and passes after the reference patch. All nine were green before the experiment began, so a failure would mean the model failed, not that the task was impossible.

2.2 Held-constant conditions and statistics

Conditions were held byte-identical: the same harness instruction files, the same prompts, and the same deterministic verify gate deciding pass or fail. Every transcript and diff was then stripped of identifying markers and scored *blind* by two separate judges—a Claude judge and a GPT judge—on a four-dimension rubric. We ran two judges precisely because a single-model judge is a known failure mode (Section 6).

Because the per-cell sample sizes are small, the statistics are *exact*, not large-sample approximations:

- **Wilson score intervals** for every pass rate.
- **McNemar's exact test** for paired head-to-heads.
- An **exact sign-flip permutation test** on tokens.
- An explicit **power statement** for what the sample can detect.

3 What 154 Trials Actually Showed

Sweep 1 was 120 passes out of 120 attempts, across both models and every effort level. Folding in the solo runs from the second sweep—the harnessed tasks and the bare-plus-harnessed greenfield build—the total is **152 of 152 clean passes** on every configuration where a model worked alone.

Table 1: Per-configuration medians, $n = 24$ per row (Actual Intelligence Labs, model-gauntlet sweep 1). Our own experiment, not externally published.

Configuration	Pass	Output tok	Input tok	Wall	Turns
Claude Fable 5 (medium effort)	24/24	1,736	19,412	68.2 s	7
Claude Fable 5 (high effort)	24/24	2,008	19,398	58.8 s	6
GPT-5.6 Sol (low effort)	24/24	1,095	95,248	31.9 s	1
GPT-5.6 Sol (medium effort)	24/24	1,534	138,252	49.0 s	1
GPT-5.6 Sol (high effort)	24/24	1,966	174,662	63.7 s	1

The honest way to report a perfect score is with its confidence interval, not as a bare 100 percent. Twenty-four passes out of twenty-four gives a 95% Wilson interval of **86.2% to 100%**. That is the correct reading: we did not prove the model never fails on tasks like these. We proved that if it has a failure rate, the data are consistent with it being anywhere from zero up to about 14 percent. A perfect sample is evidence, not a guarantee, and the interval is where the honesty lives.

For the head-to-head we used McNemar's exact test, which only extracts information from discordant pairs—the cases where one model passed and the other failed on the same task and repetition. **There were zero discordant pairs.** Every matched pair was pass and pass. With nothing discordant, there is nothing to test, and the correct conclusion is *no detectable difference between the models on this suite*. Not that they are equal—no detectable difference. Those are different claims, and we make only the one the data supports.

One configuration did wobble, and it deserves the careful version rather than a headline. We also tested a hybrid: Claude orchestrating GPT as the implementer. Across four attempts, two produced planning notes and zero code, and two shipped working solutions of roughly 160 to 176 lines—so, 2 of 4. The rig was identical across all four, so the most defensible reading is genuine run-to-run variance in a multi-agent handoff, not a fixed defect. With $n = 4$, that is where we stop. It is a flag for the next experiment, not a conclusion.

4 Does Higher Reasoning Effort Improve Code Quality?

Every model here exposes a reasoning-effort setting, and the pitch is that higher effort makes the model think harder and do better. On this suite, it did not. Pass rate was 100 percent at every effort level for both models. What moved was cost:

- Claude at high effort spent about 16 percent more output tokens than at medium (2,008 versus 1,736 median) for the identical result.
- GPT-5.6 at high effort spent about 80 percent more output tokens than at low (1,966 versus 1,095) and pulled roughly 1.8× the input tokens (174.7k versus 95.2k).

The pairwise McNemar tests between adjacent effort rungs again found zero discordant pairs. More effort, same outcome, higher bill. This matches a growing body of work on the *overthinking* phenomenon: the survey of Sui et al. [2] catalogs how reasoning models generate verbose, redundant chains that add computational cost without adding correctness, and the empirical study of Su et al. [3] is sharper still—models overthink easy problems precisely because they are poor at gauging problem difficulty. Our tasks were, in hindsight, easy for these models. Cranking the effort dial on an easy task is spending money to watch a model deliberate over a conclusion it already had.

5 Do Claude and GPT Solve Problems the Same Way?

No, and the trace makes it obvious. Claude *iterates*: a median of 6 to 7 turns per task, roughly 59 to 68 seconds of wall-clock time, reading, editing, checking, and editing again until it converges. GPT *one-shots*: a median of a single turn, roughly 26 to 64 seconds, reasoning internally and emitting a solution in one pass. Neither style won on outcomes at this tier, because both styles cleared the tier. Turn count is a workflow signature, not a quality signal.

Caveat you cannot skip: input tokens do not compare across vendors.

The config table shows Claude reporting around 19.4k median input tokens while GPT reports 95k to 175k. That is *not* GPT reading five to nine times more of your code. The two CLIs count context resends and cache reads differently, so their input-token numbers measure different things under the same label. Within-model comparisons are safe, because each CLI is consistent with itself; cross-model input-token or dollar comparisons are indicative only, and we refuse to publish them as hard claims. The axes that are honestly comparable across models are **output tokens, turns, and wall-clock time**.

6 Can You Trust an AI to Judge Another AI's Code?

Only with guardrails. Automated judges carry a documented *self-preference bias*: Wataoka et al. [4] show a model systematically favoring certain outputs when it grades, apparently because it rewards text that is more familiar—lower-perplexity—to itself. A single-model judge grading its own model's work is a conflict of interest baked into the evaluator.

So we stripped identity from every transcript and had both models grade everything. Across 139 dual-judged runs, the two judges landed within one point of each other on **96 percent** of runs, with a mean absolute gap of 0.53 points on a ten-point scale and a Pearson correlation of 0.76. On the pilot, the GPT judge ran about half a point more generous than the Claude judge on the same diffs, and both judges leaned slightly toward GPT's code. That last detail is exactly why the judging is blind and dual, and exactly why we report the disagreement instead of averaging it away.

7 What This Experiment Cannot Tell You

This section is the credibility, so we are blunt about the limits.

1. **The sample is small per arm.** With 24 trials per arm, the design has 80 percent power to detect only large gaps—roughly 36.7 percentage points or more at a 50 percent baseline. A smaller real difference would not show up; it gets a confidence interval, not a verdict.
2. **It is a ceiling effect.** Because both models passed everything, this suite cannot rank them. Reading 154 perfect runs as "the models are identical" over-reads it. The correct claim is narrower.
3. **The judges are models, not humans.** Blind, dual, and with disagreement reported, but still automated graders with their own biases.
4. **One run was re-run.** A single Claude greenfield run hit a transient CLI error (2.3 seconds, zero tokens) and was re-run. Disclosed.
5. **The tasks came from an AI-assisted pipeline.** They were proven solvable, but their difficulty was mis-calibrated low. That miscalibration is not a footnote—it is the headline finding.
6. **Cost figures use list-price placeholders.** We do not publish dollar claims off vendor-incomparable token counts.
7. **One model may have operated in the "dumb zone."** Practitioners who run these models at scale observe that output quality degrades once context grows past roughly 100k tokens, a pattern consistent with published long-context research [5]. By its own CLI's accounting, GPT-5.6 exceeded 100k cumulative input tokens on 65 of its 88 runs (median 142k, max 353k); Claude peaked at 19.6k. Two honest qualifiers: `codex` counts context resends across internal steps, so live context occupancy at any moment is not observable from our data; and the bias runs *against* GPT, which passed everything anyway. The tie is therefore safe, but the blind-judge quality comparisons carry this asymmetry, and it is on the record.

8 What Happens in the Next Experiment

The most useful result of a ceiling is that it tells you exactly where to point the next experiment. Both models beat this tier, so the tier was too easy, so the tier goes up. Experiment 2 is the hard tier: the same rig, the same blocking, the same blind dual-judge, the same exact statistics, and harder tasks pushed up in difficulty until failures start appearing. Failures are what you need. A gap only becomes measurable once at least one model starts missing, and the effort dial only earns its cost once thinking harder changes an outcome instead of just a token count. Same exam room, harder exam. That is where Claude versus GPT gets a real answer—and we will report that one the same way we reported this one: with the intervals drawn in, the disagreements shown, and no winner invented that the data did not earn.

References

[1] N. Jain et al., *LiveCodeBench: Holistic and Contamination-Free Evaluation of Large Language Models for Code*, 2024. https://arxiv.org/abs/2403.07974

[2] Y. Sui et al., *Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models*, 2025. https://arxiv.org/abs/2503.16419

[3] J. Su et al., *Between Underthinking and Overthinking: An Empirical Study of Reasoning Length*, 2025. https://arxiv.org/abs/2505.00127

[4] K. Wataoka et al., *Self-Preference Bias in LLM-as-a-Judge*, 2024. https://arxiv.org/abs/2410.21819

[5] N. F. Liu et al., *Lost in the Middle: How Language Models Use Long Contexts*, 2024. https://arxiv.org/abs/2307.03172