
Consistency Amplifies: How Behavioral Variance Shapes Agent Accuracy

Aman Mehta¹

Abstract

As LLM-based agents are deployed in production systems, understanding their behavioral consistency (whether they produce similar action sequences when given identical tasks) becomes critical for reliability. We study consistency in the context of SWE-bench, a challenging software engineering benchmark requiring complex, multi-step reasoning. Comparing Claude 4.5 Sonnet, GPT-5, and Llama-3.1-70B across 50 runs each (10 tasks \times 5 runs), we find that across models, higher consistency aligns with higher accuracy: Claude achieves the lowest variance (CV: 15.2%) and highest accuracy (58%), GPT-5 is intermediate (CV: 32.2%, accuracy: 32%), and Llama shows the highest variance (CV: 47.0%) with lowest accuracy (4%). However, within a model, consistency can amplify both correct and incorrect interpretations. Our analysis reveals a critical nuance: **consistency amplifies outcomes rather than guaranteeing correctness**. 71% of Claude’s failures stem from “consistent wrong interpretation”: making the same incorrect assumption across all runs. Interestingly, GPT-5 achieves similar early strategic agreement as Claude (diverging at step 3.4 vs 3.2) but exhibits $2.1\times$ higher variance, suggesting that divergence timing alone does not determine consistency. These findings suggest that for production deployment, interpretation accuracy matters more than execution consistency, with implications for agent evaluation and training.

1. Introduction

The deployment of LLM-based agents in production systems (from code assistants to autonomous research tools) has accelerated dramatically. Yet a fundamental question remains underexplored: when given the same task multiple

times, do these agents behave consistently?

Behavioral consistency matters for several reasons. First, inconsistent agents are unpredictable, making them difficult to trust in high-stakes applications. Second, variance in agent behavior complicates debugging and improvement; if an agent sometimes succeeds and sometimes fails on identical inputs, isolating the cause becomes challenging. Third, consistency is a prerequisite for meaningful benchmarking: if results vary substantially across runs, single-run evaluations may be misleading.

Prior work has established that LLMs exhibit significant variance in simple reasoning tasks (Mehta, 2026). However, the relationship between consistency and task complexity remains unclear. Do consistency challenges grow with action space size? Does the variance in simple tasks predict variance in complex ones?

We investigate these questions using SWE-bench (Jimenez et al., 2023), a benchmark requiring agents to resolve real GitHub issues through multi-step code modifications. SWE-bench tasks demand exploration of large codebases, understanding of bug context, implementation of fixes, and verification, representing a substantial increase in complexity from prior consistency studies.

Our contributions are:

1. **Quantitative characterization across capability tiers:** We find that consistency aligns with accuracy across three models: Claude 4.5 Sonnet (CV: 15.2%, accuracy: 58%), GPT-5 (CV: 32.2%, accuracy: 32%), and Llama-3.1-70B (CV: 47.0%, accuracy: 4%).
2. **The amplification insight:** We show that consistency amplifies outcomes rather than guaranteeing correctness. 71% of Claude’s failures are “consistent wrong interpretation”: the same incorrect approach across all runs.
3. **The speed-accuracy-consistency tradeoff:** GPT-5 is $4.7\times$ faster than Claude (9.9 vs 46.1 steps) but achieves $1.8\times$ lower accuracy and $2.1\times$ worse consistency, revealing a fundamental tradeoff.
4. **The fixation failure mode:** Through detailed trajec-

¹Snowflake Inc.. Correspondence to: Aman Mehta <aman.mehta@snowflake.com>.

tory analysis, we identify cases where Claude’s thoroughness causes fixation on incorrect interpretations, while less thorough approaches occasionally allow course correction.

5. **Divergence timing vs consistency:** Despite similar divergence timing (Claude: step 3.2, GPT-5: step 3.4), Claude achieves $2.1 \times$ better consistency, showing that early agreement alone does not determine behavioral variance.

These findings have implications for agent deployment: consistency alone is insufficient for reliability; interpretation quality is the bottleneck.

2. Related Work

LLM Agent Benchmarks. SWE-bench (Jimenez et al., 2023) evaluates agents on real GitHub issues, requiring multi-step reasoning and code modification. Other benchmarks include WebArena (Zhou et al., 2023) for web navigation, OSWorld (Xie et al., 2024) for computer use, and various coding benchmarks (Chen et al., 2021; Austin et al., 2021). Most report single-run accuracy, leaving consistency unexplored. Recent work on SWE-bench Verified (Jimenez et al., 2024) provides human-validated solutions, enabling more reliable evaluation.

LLM Consistency and Reliability. Prior work has examined consistency in simpler settings: mathematical reasoning (Wang et al., 2023), factual questions (Elazar et al., 2021), and multi-hop QA (Mehta, 2026). These studies find substantial variance even in straightforward tasks. Self-consistency methods (Wang et al., 2023) leverage this variance through majority voting, but assume independent samples, an assumption that may not hold for complex agent trajectories. We extend this line of work to complex, multi-step agent behavior where actions are sequentially dependent.

Agent Failure Analysis. Studies of agent failures have focused on error categorization (Liu et al., 2023) and recovery mechanisms (Shinn et al., 2023). Reflexion (Shinn et al., 2023) shows that agents can improve through verbal self-reflection, but assumes failures are detectable. Our work contributes a new perspective: systematic analysis of *when* in the trajectory failures originate, and how consistency interacts with correctness. We find that many failures are “consistent wrong”; the agent confidently repeats the same mistake, making reflection-based recovery unlikely.

Table 1. Task characteristics. Tasks vary in bug type, fix complexity (lines changed), and file count.

Task ID	Bug Type	Fix Size	Files
12907	Logic error	3 lines	1
13033	Missing check	5 lines	1
13236	Silent conversion	4 lines	1
13398	Edge case	8 lines	2
13453	Type handling	6 lines	1
13579	Format string	2 lines	1
13977	Deprecation	12 lines	2
14096	Boundary check	7 lines	1
14182	Default value	4 lines	1
14309	Import error	3 lines	1

3. Experimental Setup

3.1. Benchmark and Tasks

We use SWE-bench Verified, a curated subset of SWE-bench with human-validated solutions. We select 10 tasks from the astropy repository, chosen for diversity in bug types and fix complexity. Table 1 summarizes the task characteristics.

Each task requires the agent to understand a GitHub issue, locate relevant code, implement a fix, and verify correctness. The median fix size is 4.5 lines, but finding the right location and understanding the bug context requires substantial exploration.

3.2. Models

We compare three models representing different capability levels:

- **Claude 4.5 Sonnet** (claude-sonnet-4-5-20250929): A frontier model known for strong coding capabilities, with extended context and tool use.
- **GPT-5** (openai-gpt-5, February 2026): OpenAI’s frontier model with strong reasoning capabilities, available through the OpenAI API.
- **Llama-3.1-70B-Instruct**: An open-weights model, smaller but widely deployed in production systems.

All models have access to identical tools: bash commands for file system navigation, code editing, and test execution. We use the same system prompt and tool definitions for all.

Agent Framework. We use mini-SWE-agent (Jimenez et al., 2023), a minimal agent scaffold that provides only a bash interface (no tool-calling API). Each action is executed via `subprocess.run` in an isolated Docker container. During development, we identified and fixed a format-error

bug where agents could inadvertently submit during error recovery; patch submission is now blocked during format-error handling. Our framework and fix are available in the supplementary code.

3.3. Experimental Protocol

For each model-task pair, we run 5 independent trials with:

- Temperature: 0.5 (moderate stochasticity)
- Maximum steps: 250
- Isolated Docker containers per run
- Identical prompts and tool access
- Fresh repository state for each run

This yields 50 runs per model (10 tasks \times 5 runs), totaling 150 agent trajectories across three models. We set a per-run cost limit of \$10 (estimated from API token pricing); no run was truncated by this budget—the maximum observed cost was \$2.91 (Claude). All runs completed without infrastructure failures or budget truncation.

3.4. Metrics

Consistency. We measure behavioral variance using the coefficient of variation (CV) of step counts:

$$CV = \frac{\sigma_{\text{steps}}}{\mu_{\text{steps}}} \times 100\% \quad (1)$$

Lower CV indicates more consistent behavior. We compute CV per task and aggregate across tasks. We also measure sequence diversity: the fraction of runs with unique action sequences.

Accuracy. We evaluate patch correctness using the official SWE-bench evaluation harness, which applies each submitted patch to an isolated Docker container and runs the repository’s test suite (approximately 110 seconds per task). A patch is marked “resolved” only if all previously-failing tests pass. This is distinct from agent completion rate (whether a patch was submitted), which approaches 100% for all models.

Phase Decomposition. We categorize each agent action into phases based on command type:

- **EXPLORE:** Directory listing, file search (`ls`, `find`, `grep`)
- **UNDERSTAND:** File reading (`cat`, `head`, `less`)
- **EDIT:** Code modification (`sed`, `echo >>`, `file writes`)

Table 2. Overall comparison across three models on SWE-bench (10 tasks, 5 runs each, 50 runs per model).

Metric	Claude	GPT-5	Llama
Consistency (CV)	15.2%	32.2%	47.0%
Accuracy	58%	32%	4%
Avg Steps	46.1	9.9	17.0
Cost/Run [†]	\$1.50	\$0.53	\$0.10
Valid Patches	50/50	48/50	40/50
Unique Sequences	100%	100%	100%

[†]API inference cost only; excludes SWE-bench evaluation harness compute.

- **VERIFY:** Testing (`python`, `pytest`)

This decomposition allows us to identify where variance originates in the agent trajectory.

4. Results

4.1. RQ1: Do Models Differ in Consistency?

Table 2 presents our main findings. The three models show a clear hierarchy in both consistency and accuracy.

The three models show a clear hierarchy. Since each model was evaluated on the same 10 tasks, we use paired t -tests ($df = 9$) for all cross-model comparisons. Claude is significantly more consistent than GPT-5 ($t(9) = 2.59$, $p = 0.029$, Cohen’s $d = 1.16$), which trends more consistent than Llama ($t(9) = -1.92$, $p = 0.087$). Claude’s CV of 15.2% indicates step counts vary by about 7 steps around a mean of 46, while GPT-5’s 32.2% CV indicates variation of about 3 steps around a mean of 10.

Notably, **100% of runs produce unique action sequences** for all three models. Even with temperature 0.5, no two runs follow identical paths. This highlights that behavioral consistency (low CV) does not mean deterministic behavior; models can be consistent in *strategy* while varying in *execution details*.

The Speed-Accuracy-Consistency Tradeoff. GPT-5 reveals an important tradeoff: it is $4.7\times$ faster than Claude (9.9 vs 46.1 steps) but achieves $1.8\times$ lower accuracy (32% vs 58%) and $2.1\times$ worse consistency (CV: 32.2% vs 15.2%). This suggests that thoroughness trades off against speed, with consistency as a mediating factor.

Figure 1 visualizes the consistency difference across all three models, and Figure 2 shows the raw step counts across all 150 runs. The heatmap makes the consistency gap visually striking: Claude’s uniform coloring indicates similar step counts across runs, while Llama’s patchy appearance reflects high variance.

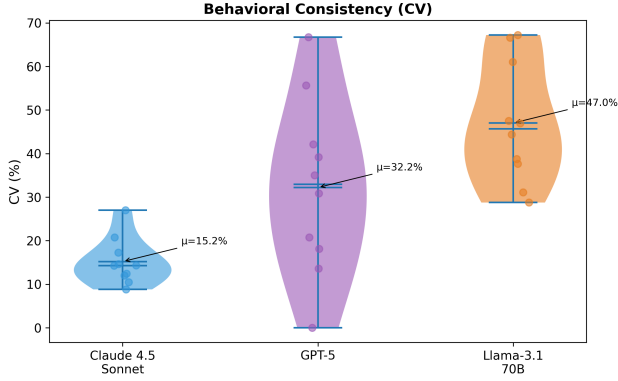


Figure 1. Behavioral consistency comparison across three models. Claude achieves the lowest coefficient of variation, followed by GPT-5, then Llama. Individual task CVs shown as points.



Figure 2. Step count heatmap across all 150 runs. Claude (left) shows uniform coloring; GPT-5 (middle) shows low counts with moderate variance; Llama (right) shows high variance.

4.2. RQ2: Does Consistency Relate to Accuracy?

With three models, we can examine whether consistency predicts accuracy across capability levels. The rank correlation is perfect: Claude (most consistent, most accurate), GPT-5 (middle), Llama (least consistent, least accurate).

However, we cannot establish causality from $n = 3$ models. To investigate further, we examine *within-model* relationships.

Within-Model Analysis. We compute the correlation between per-task CV and accuracy for each model separately. Surprisingly, we find **no significant correlation** within any model:

- Claude: $r = -0.10, p = 0.78$
- GPT-5: $r = -0.15, p = 0.68$
- Llama: $r = 0.30, p = 0.40$

Figure 3 visualizes this relationship. This suggests consistency and accuracy are not straightforwardly linked at the task level; a model can be consistently wrong or inconsistently right.

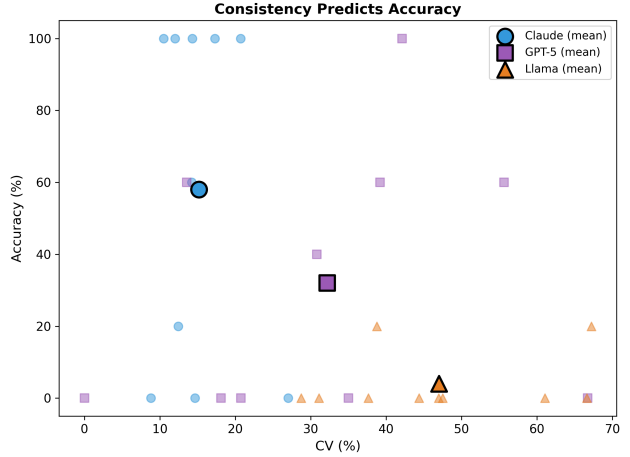


Figure 3. Per-task consistency (CV) vs accuracy across three models. No significant within-model correlation exists. Consistency does not predict accuracy at the task level.

Table 3. Consistency amplifies outcomes: Claude’s performance by interpretation quality. Tasks are categorized by whether Claude correctly understood the bug.

Pattern	Tasks	Runs	Accuracy
Correct interpretation	5	25	100% (25/25)
Wrong interpretation	3	15	0% (0/15)
Mixed results	2	10	40% (4/10)

The Amplification Insight. Closer analysis reveals why: consistency *amplifies* outcomes rather than determining them. Table 3 shows that Claude’s tasks fall into distinct patterns.

When Claude correctly interprets a task, it succeeds on **all 5 runs**. When it incorrectly interprets a task, it fails on **all 5 runs**. Consistency ensures that correct approaches are reliably executed, but also that incorrect approaches are reliably repeated.

Cross-Model Task Performance. Table 4 shows per-task accuracy across all three models. Notably, GPT-5 solves 6/10 tasks at least once (vs Claude 7/10, Llama 2/10), and achieves perfect 5/5 accuracy on astropy-14309, matching Claude.

Figure 4 shows the per-task accuracy comparison.

4.3. RQ3: Where Does Variance Originate?

We decompose agent trajectories into phases to identify variance sources. Table 5 shows the distribution across all three models.

GPT-5’s Unique Profile. GPT-5 shows a distinctive pattern: it spends the most time on VERIFY (32.3%), aggres-

Table 4. Per-task accuracy across models. GPT-5 solves 6/10 tasks at least once.

Task	Claude	GPT-5	Llama
12907	5/5	2/5	0/5
13033	3/5	0/5	0/5
13236	0/5	0/5	1/5
13398	0/5	0/5	0/5
13453	5/5	3/5	0/5
13579	5/5	3/5	0/5
13977	0/5	0/5	0/5
14096	5/5	3/5	0/5
14182	1/5	0/5	0/5
14309	5/5	5/5	1/5
Total	29/50	16/50	2/50

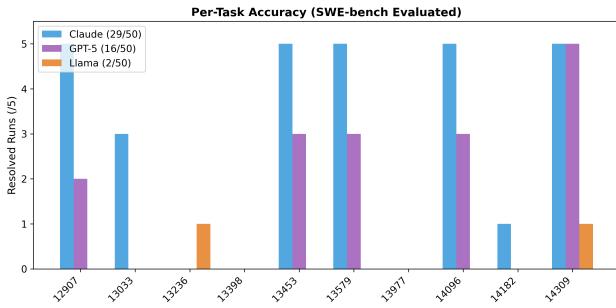


Figure 4. Per-task accuracy comparison across three models. Claude achieves 58% overall, GPT-5 32%, and Llama 4%. Note that Llama outperforms both on astropy-13236.

sively testing with `python` commands. It also uses `nl` (numbered line output) heavily (19.4% of actions), a command rarely used by other models. Despite less exploration (13.0% vs Claude’s 17.8%), GPT-5 achieves moderate accuracy through rapid iteration.

Figure 5 visualizes this decomposition.

Variance by Phase. Claude shows the lowest within-phase variance across all phases, while Llama shows erratic behavior especially in EXPLORE (CV: 123% vs Claude’s 42%). GPT-5 falls between, with notably low variance in VERIFY due to its consistent testing strategy.

4.4. RQ4: How Do Failure Modes Differ?

We categorize all failures by type. Table 6 shows that all three models primarily fail by submitting incorrect fixes rather than giving up.

Claude and GPT-5 almost never give up (0% and 6% empty patches), while Llama submits empty patches 21% of the time. This “give up” failure mode is largely absent from the more capable models.

Figure 6 visualizes these distributions.

Table 5. Phase decomposition showing where models spend their steps (% of total actions).

Phase	Claude	GPT-5	Llama
EXPLORE	17.8%	13.0%	28.1%
UNDERSTAND	41.2%	31.4%	30.5%
EDIT	14.5%	12.0%	11.2%
VERIFY	19.3%	32.3%	18.9%
OTHER	7.2%	11.3%	11.3%
Pre-edit	59.0%	44.4%	58.6%

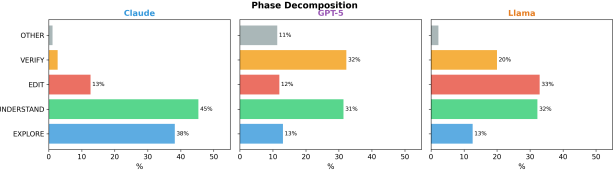


Figure 5. Phase decomposition across three models. Claude invests heavily in UNDERSTAND, GPT-5 emphasizes VERIFY, and Llama spends more on EXPLORE.

The Consistent Wrong Pattern. We further analyze Claude’s 21 failures and find that 71% (15/21) are “consistent wrong interpretation”: the same incorrect approach across all 5 runs of a task. On three tasks (astropy-13236, astropy-13398, astropy-13977), Claude makes the identical mistake every time.

This reveals a critical insight: **interpretation failures dominate execution failures**. More testing cannot help if the fundamental understanding is wrong.

Annotation Protocol. To classify failures as “consistent wrong interpretation” vs. “execution error,” the first author examined each failed run’s trajectory and final patch. A failure was labeled “interpretation error” if the agent’s patch addressed a different semantic goal than the ground-truth fix (e.g., adding a deprecation warning instead of removing code). “Execution error” was assigned when the patch targeted the correct goal but contained implementation bugs. This labeling was not blinded; we note it as a limitation and release all trajectories for independent verification.

4.5. RQ5: When Do Trajectories Diverge?

Having established that models differ in consistency (RQ1) and identified phase-level patterns (RQ3), we investigate the mechanism: at which step do runs of the same task first take different actions?

Divergence Analysis. We define the divergence step as the first step where not all 5 runs of a task execute the same action category. Table 7 reveals a surprising finding: Claude and GPT-5 have nearly identical divergence timing, yet very

Table 6. Failure mode taxonomy across three models. All models primarily fail via wrong fixes.

Failure Mode	Claude (21)	GPT-5 (34)	Llama (48)
WRONG_FIX	21 (100%)	32 (94%)	38 (79%)
EMPTY_PATCH	0 (0%)	2 (6%)	10 (21%)
LOOP_DEATH	0 (0%)	0 (0%)	0 (0%)

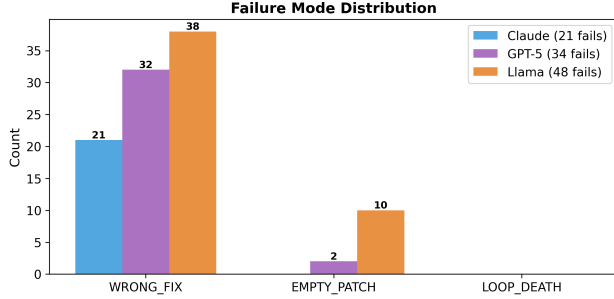


Figure 6. Failure mode distribution across three models. All models primarily fail via wrong fixes; Llama uniquely shows “give up” failures.

different consistency.

The most striking finding is the two-tier pattern: Claude and GPT-5 both diverge around step 3, while **60% of Llama tasks diverge at Step 1**. Yet despite similar divergence timing, Claude achieves $2.1\times$ better consistency (CV: 15.2% vs 32.2%). This suggests that **early strategic agreement is necessary but not sufficient for consistency**; what happens after divergence matters equally.

First Action Analysis. We next ask whether the first action predicts success. Table 8 shows that first action strongly predicts *model* but not *success*.

GPT-5’s 100% 1s first action is the most predictable opening of any model, yet it achieves only 32% accuracy. Meanwhile, Claude with the same 1s start succeeds 85% of the time. This confirms that **strategic coherence, not starting action, predicts success**.

Interpretation. These analyses reveal that early agreement can be double-edged: when Claude commits to a wrong interpretation within its coherent first steps, all runs fail identically (the “consistent wrong” pattern from RQ4).

5. Case Studies

To understand our quantitative findings more deeply, we examine two contrasting cases.

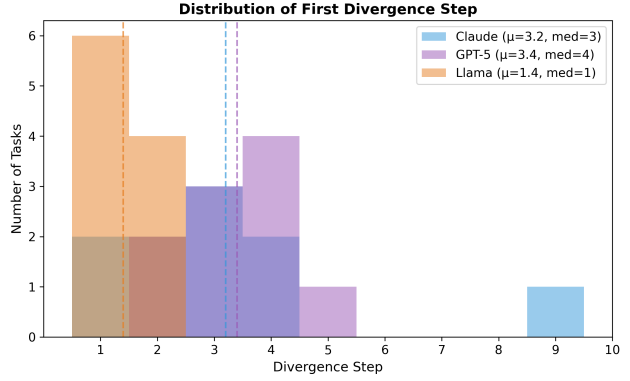


Figure 7. Divergence step distribution. Claude and GPT-5 diverge around step 3; Llama diverges immediately (60% at Step 1).

Table 7. Divergence analysis: Claude and GPT-5 diverge at similar steps, but Claude achieves $2.1\times$ better consistency.

Model	Mean Div.	Median	% at Step 1	% by Step 5
Claude	3.2	3.0	0%	90%
GPT-5	3.4	3.5	0%	100%
Llama	1.4	1.0	60%	100%

5.1. Case 1: When Thoroughness Backfires (astropy-13236)

This is the only task where Llama outperformed both Claude and GPT-5 (20% vs 0% vs 0%).

The Bug. The issue reports that adding a structured numpy array to an Astropy Table silently converts it to NdataArrayMixin, losing functionality. The expected fix: remove the automatic conversion (4 lines of code).

What Happened. Claude interpreted the task as “add a deprecation warning but preserve existing behavior.” It spent 30–50 steps per run implementing and debugging a FutureWarning. All 5 runs failed because the tests expected the behavior to be *removed*, not deprecated. GPT-5 made a similar interpretation error but with fewer steps (7 per run), failing equally consistently.

Llama, in its one successful run, interpreted the task correctly: “remove the conversion code.” It made the change in 13 steps and passed the tests.

The Fixation Failure Mode. Claude’s thoroughness, usually an asset, became a liability: it never questioned its initial interpretation. GPT-5’s speed meant it failed faster but equally consistently. Llama’s variance occasionally stumbled onto the correct interpretation.

Table 8. First action distribution. GPT-5 always starts with `ls` (100%), yet achieves only 32% accuracy.

First Action	Claude	GPT-5	Llama
<code>find</code>	68%	0%	20%
<code>ls</code>	26%	100%	54%
<code>grep</code>	2%	0%	24%
<code>cat</code>	4%	0%	2%

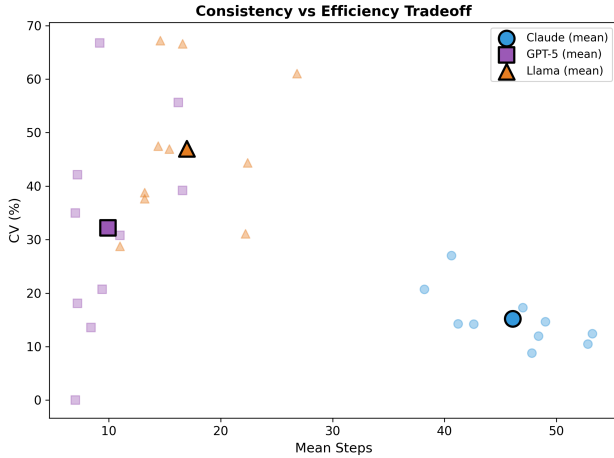


Figure 8. Efficiency-consistency tradeoff across three models. Claude (blue) clusters in high-step, low-CV region. GPT-5 (green) shows low steps with moderate CV. Llama (orange) shows high variance regardless of step count.

5.2. Case 2: The Efficiency Paradox (astropy-14309)

This is the only task where all three models achieved at least one success, and the only task where GPT-5 matched Claude’s perfect 5/5 accuracy.

The Bug. A simple import error where a function was not properly exposed in the module’s `__init__.py`.

What Happened. Claude solved it 5/5 times with mean 53.2 steps. GPT-5 also solved it 5/5 times but with only **7 steps** on average. Llama solved it 1/5 times with 8 steps in the successful run.

Analysis. For this simple task, GPT-5’s quick approach (7 steps) matched Claude’s thorough approach (53 steps) in accuracy while being $7.6\times$ faster. This reveals the efficiency paradox: thoroughness provides robustness for complex tasks but may be unnecessary for simple ones. An optimal agent might adapt its strategy based on estimated task complexity.

Figure 8 visualizes this tradeoff across all tasks.

6. Discussion

6.1. The Interpretation Bottleneck

Our findings suggest that for complex agentic tasks, the bottleneck is interpretation rather than execution. Claude executes its plans consistently and thoroughly, but 71% of its failures stem from incorrect initial interpretation. GPT-5, despite being faster, shows the same pattern.

This has implications for agent design. Current approaches focus on execution quality: better tool use, more thorough testing, longer trajectories. Our results suggest that improving initial task interpretation may yield larger gains.

Interpretation is Necessary but Not Sufficient. While interpretation failures dominate our dataset (71% of Claude’s failures), two tasks (astropy-13398 and astropy-13977) reveal a complementary failure mode: all three models correctly identified the required change but failed to implement it. Task 13398 required new coordinate transformation mathematics; task 13977 required surgically precise edits with subtle edge cases. These “implementation failures” (0/15 across all models) suggest that some tasks exceed current model capabilities regardless of interpretation quality. Our main finding—that interpretation is the primary bottleneck—holds for tasks within capability range, but capability ceilings remain a distinct limiting factor.

6.2. Consistency as Double-Edged Sword

The common assumption that “more consistent = more reliable” requires nuance. Consistency is valuable when the approach is correct, as it ensures reliable execution. But consistency also means reliable repetition of errors.

Our three-model comparison strengthens this insight: the rank ordering (Claude > GPT-5 > Llama) holds for both consistency and accuracy, but the relationship is not causal at the task level.

6.3. The Speed-Accuracy-Consistency Triangle

GPT-5 reveals a fundamental tradeoff:

- $4.7\times$ faster than Claude (9.9 vs 46.1 steps)
- $1.8\times$ lower accuracy (32% vs 58%)
- $2.1\times$ worse consistency (CV: 32.2% vs 15.2%)

Practitioners must choose their priority based on deployment context: rapid prototyping may favor GPT-5’s speed, while production systems may require Claude’s reliability.

6.4. Divergence Timing is Not Enough

Perhaps our most surprising finding is that Claude and GPT-5 diverge at nearly identical steps (3.2 vs 3.4) yet achieve very different consistency (CV: 15.2% vs 32.2%). This suggests that early strategic agreement, while necessary, is not sufficient. What happens *after* divergence matters: Claude maintains coherent strategies across runs even after initial divergence, while GPT-5’s trajectories scatter more widely.

6.5. Implications for Benchmarking

Our finding that 100% of runs produce unique sequences has implications for how we evaluate agents. Standard practice reports single-run accuracy, but our results show this may be misleading:

- A 60% accuracy score could mean: 60% of tasks solved consistently, or 100% of tasks solved 60% of the time
- These have different implications for deployment reliability
- Multi-run evaluation with consistency reporting should become standard

7. Limitations

Sample Size. With $n = 3$ models and 10 tasks, we can identify trends but cannot establish strong statistical relationships. A larger study with 5+ models would enable regression analysis.

Single Domain. All tasks are from the astropy repository. Different codebases, languages, or bug types may show different patterns. Astropy is a well-documented, mature codebase; messier codebases might show different results.

Temperature. We use temperature 0.5 throughout. Different temperatures would likely change both consistency and accuracy, and the relationship between them.

Causality. We cannot prove that consistency causes accuracy. An intervention study (forcing models to explore more or less) would strengthen causal claims. We observe correlation and provide mechanistic explanations, but cannot rule out confounds.

8. Conclusion

We studied behavioral consistency in LLM agents on SWE-bench across three models, finding a clear hierarchy: Claude achieves the highest consistency (CV: 15.2%) and accuracy

(58%), GPT-5 is intermediate (CV: 32.2%, accuracy: 32%), and Llama shows the highest variance (CV: 47.0%) with lowest accuracy (4%).

Our key insight is that consistency amplifies outcomes rather than guaranteeing correctness: 71% of Claude’s failures are “consistent wrong interpretation.” We also discovered that divergence timing alone does not determine consistency; Claude and GPT-5 diverge at similar steps but achieve very different variance.

GPT-5 reveals a speed-accuracy-consistency tradeoff: it is $4.7\times$ faster than Claude but achieves $1.8\times$ lower accuracy and $2.1\times$ worse consistency. These findings suggest that interpretation quality, not execution consistency, is the primary bottleneck for reliable agent deployment. Future work should explore adaptive strategies that balance thoroughness with efficiency based on task complexity.

References

- Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Elazar, Y., Kassner, N., Ravfogel, S., Ravichander, A., Hovy, E., Schütze, H., and Goldberg, Y. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9: 1012–1031, 2021.
- Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.
- Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench verified: A human-validated subset for reliable agent evaluation. *OpenAI Blog*, 2024. Available at: <https://openai.com/index/introducing-swe-bench-verified/>.
- Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.
- Mehta, A. When agents disagree with themselves: Measuring behavioral consistency in llm-based agents. *arXiv preprint arXiv:2602.11619*, 2026.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shi, D., Lu, Z., et al. Os-world: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A. Per-Task Results

Table 9 shows detailed per-task results for all three models.

B. Trajectory Examples

We provide example trajectories illustrating key findings. Full trajectories are available in our code repository.¹

Claude’s Consistent Correct Approach (astropy-13453). Claude follows a systematic pattern: explore directory structure (8 steps), read relevant files (12 steps), implement fix (6 steps), verify with tests (5 steps). All 5 runs follow nearly identical sequences, varying only in exploration order.

GPT-5’s Fast Iteration (astropy-14309). GPT-5 solves this task in just 7 steps on average: `ls` → `nl` → `grep` → `sed` → `python`. Its heavy use of `nl` (numbered line output) is distinctive.

Llama’s Lucky Path (astropy-14309). Run 4 succeeds in just 8 steps: `ls` → `grep` → `cat` → `sed` → `python test.py`. The direct path happens to hit the right file immediately. Other runs explore different (wrong) paths first.

C. Statistical Details

CV Comparisons. Since models are evaluated on the same 10 tasks, we use *paired t*-tests ($df = 9$) for CV and divergence comparisons:

- GPT-5 vs Claude: $t(9) = 2.59, p = 0.029, d = 1.16$ (significant)
- GPT-5 vs Llama: $t(9) = -1.92, p = 0.087, d = -0.86$ (not significant)
- Claude vs Llama: $t(9) = -6.75, p < 0.001, d = -3.02$ (significant)

Accuracy Comparisons. Fisher’s exact test (accuracy is binary per run, not naturally paired):

- GPT-5 vs Claude: $OR = 0.34, p = 0.015$ (significant)
- GPT-5 vs Llama: $OR = 11.3, p < 0.001$ (significant)
- Claude vs Llama: $OR = 33.1, p < 0.001$ (significant)

Divergence Comparisons. Paired *t*-tests ($df = 9$):

- GPT-5 vs Claude: $t(9) = 0.25, p = 0.808$ (not significant)
- GPT-5 vs Llama: $t(9) = 5.77, p < 0.001$ (significant)
- Claude vs Llama: $t(9) = 2.57, p = 0.030$ (significant)

Summary.

- Consistency (CV): Claude \ll GPT-5 \approx Llama
- Accuracy: Claude \gg GPT-5 \gg Llama (all pairwise significant)
- Divergence: Claude \approx GPT-5 \gg Llama

¹<https://github.com/amanmehta-maniac/agent-consistency>

Table 9. Per-task breakdown of consistency (CV%), accuracy, and mean steps for all three models.

Task	Claude			GPT-5			Llama		
	CV	Acc	Steps	CV	Acc	Steps	CV	Acc	Steps
12907	20.7	5/5	38	30.8	2/5	11	66.6	0/5	17
13033	14.2	3/5	43	66.7	0/5	9	44.4	0/5	22
13236	27.0	0/5	41	35.0	0/5	7	38.8	1/5	13
13398	8.8	0/5	48	20.7	0/5	9	47.0	0/5	15
13453	14.3	5/5	41	39.2	3/5	17	28.7	0/5	11
13579	10.5	5/5	53	13.6	3/5	8	47.5	0/5	14
13977	14.6	0/5	49	0.0	0/5	7	37.7	0/5	13
14096	17.3	5/5	47	55.6	3/5	16	31.1	0/5	22
14182	12.4	1/5	53	18.1	0/5	7	61.1	0/5	27
14309	12.0	5/5	48	42.1	5/5	7	67.2	1/5	15
Mean	15.2	29/50	46.1	32.2	16/50	9.9	47.0	2/50	17.0