AGENTIC UNCERTAINTY REVEALS AGENTIC OVERCONFIDENCE

Jean Kaddour[1,2] **Srijan Patel**[2] **Gbètondji Dovonon**[1] **Leo Richter**[1]

Pasquale Minervini[3] **Matt Kusner**[4,5]

[1]University College London [2]SevnAI [3]University of Edinburgh
[4]Mila – Québec AI Institute [5]Polytechnique Montréal
jean.kaddour.20@ucl.ac.uk

ABSTRACT

Can AI agents predict whether they will succeed at a task? We study **agentic uncertainty** by eliciting success probability estimates before, during, and after task execution. All results exhibit **agentic overconfidence**: some agents that succeed only 22% of the time predict 77% success. Counterintuitively, pre-execution assessment with strictly less information *tends* to yield better discrimination than standard post-execution review, though differences are not always significant. Adversarial prompting reframing assessment as bug-finding achieves the best calibration. Code is available at https://github.com/sevn-ai/agentic-uncertainty.

1 INTRODUCTION

A software engineer needs to fix an auth service error. Before delegating to an AI coding agent, she asks: what are the chances this succeeds?

Pre-Exec. **P(success): 72%**

```
The issue is clear and well-defined, the error message points
directly to the problem, and the fix follows existing patterns in
the codebase.
```

72% confidence before any code is written. As the coding agent works, she asks another agent to monitor progress:

Mid-Exec. **P(success): 78%**

```
The agent has correctly diagnosed the problem and knows exactly what
code to add.  The probability of success is high.
```

Confidence *rises* to 78%. The patch is now complete. She fires off a review agent:

Post-Exec. **P(success): 92%**

```
The patch is a correct and complete fix.  It's a minimal, focused
change that adds the missing interface method.
```

Too optimistic. Let's spawn an adversarial agent.

Adv. Post-Exec. **P(success): 85%**

```
Minor concerns don't affect the main use case...  the patch correctly
resolves the reported issue.
```

Still 85%. All four agents confidently predict success.

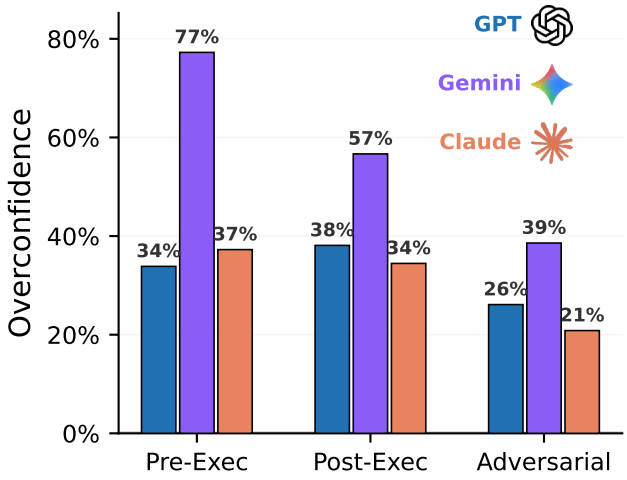

Figure 1: **Agentic overconfidence**. We measure the overconfidence as the difference between the estimated success probability and the true success probability (true rates: GPT-5.2 Codex 35%, Gemini-3-Pro 22%, Opus 4.5 27%). We plot three strategies: pre-, post-, and adversarial-post-execution. All agents systematically overestimate their success.

But the patch fails! And this **agentic overconfidence** is systematic. For example, GPT-5.2-Codex-based post-execution agents predict 73% success against a true rate of 35% averaged over 100 SWE-Bench-Pro (Deng et al., 2025) tasks.

This matters because the scope of autonomous work is expanding rapidly. The effective length of tasks that AI agents complete has doubled every 7 months for six years (METR, 2025). As we increasingly delegate complex workflows to agents (Appel et al., 2025), we must develop scalable oversight protocols (Bowman et al., 2022).

In this work, we elicit *agentic uncertainty* at three points in a coding agent's lifecycle: pre-, mid-, and post-execution. Each corresponds to a different oversight question: Can agents predict failure before committing resources? Can they recognize failure as it unfolds? Can they verify their own work? Importantly, we use the same underlying model for both the coding agent that produces patches and the uncertainty agent that assesses them, isolating the effect of information access from differences in model capability.

Our experiments on 100 SWE-bench Pro tasks across three frontier models (GPT-5.2-Codex, Gemini 3 Pro, Claude Opus 4.5) reveal several striking findings:

- **Pervasive overconfidence.** Post-execution agents can predict 73% success on average against a 35% base rate (GPT), with similar gaps across all models.
- **More context, uncalibrated doubt.** Mid-execution agents develop "cold feet": confidence decreases as they observe their partial work, but this doubt is uninformative, occurring equally for successes and failures.
- **Adversarial framing helps.** Prompting agents to "find bugs" rather than "verify correctness" reduces overconfidence by up to 15 pp and tends to achieve the best calibration across models in our setup.

2 Methods

2.1 Problem Setup

We define **agentic uncertainty** as an agent's estimate of the probability that an agent built on the same underlying model will successfully complete a task. The uncertainty (-estimating) agent may

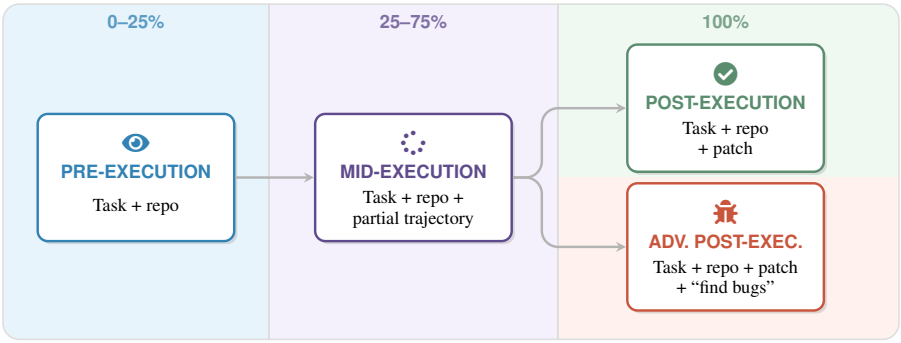

Figure 2: **Agentic uncertainty regimes.** Each regime observes different information. Post-execution and adversarial post-execution occur at the same point but use different prompts.

use a different system prompt or have access to different information than the task-solving agent, but shares the same base model.

Unlike standard uncertainty quantification, which focuses on confidence in individual predictions or token probabilities, agentic uncertainty concerns the outcome of an entire multi-step trajectory: will this sequence of observations, reasoning, and actions culminate in task success?

Kadavath et al. (2022) introduced P(IK): "probability that I know," measuring whether models can predict which questions they can answer correctly. We generalize this to agentic settings and call it **P(IS)**: "probability that I succeed." Formally, given a task t, base model M, and information state $\mathcal{I}$ available at elicitation time:

$$P(\text{IS}) \;:=\; P(\text{agent}_M \text{ succeeds on } t \mid \mathcal{I}) \tag{1}$$

where $\mathcal{I}$ may include the task description, repository state, partial trajectory, or proposed patch, depending on the elicitation regime.

2.2 Uncertainty Agents

We consider three points in an agent's life cycle to elicit uncertainty estimates: before attempting a solution, during execution, and after producing a solution.

Each uncertainty agent interacts with a sandbox environment through tool use: reading files, searching code, and inspecting version history. The key distinction is the information available at uncertainty estimation time.

Pre-Execution Agent. In the pre-execution setting, an agent receives only the task description (*e.g.*, a GitHub issue) and read-only access to the repository. Crucially, the agent cannot execute code, run tests, or modify files. The agent must form an estimate based on cues, e.g., the complexity of the codebase, rather than through trial and error.

Mid-Execution Agent. Rather than eliciting estimates at a single point, we prompt the agent at 25%, 50%, and 75% of total agent steps (tool calls and reasoning turns) to estimate the success probability given the trajectory so far.

Post-Execution Agent. Once a task solution has been proposed, can we trust an agent's self-assessment that it is correct? In the post-execution setting, an agent receives both the task description and a proposed patch. The repository is in its post-patch state, and the agent can explore the modified codebase. After assessment, the agent estimates whether the patch successfully solves the task.

Adversarial post-execution variant. We also evaluate a variant that explicitly prompts agents to find bugs before estimating confidence. Rather than asking "is this correct?", adversarial post-execution asks "what bugs can you find?" This reframes the task from verification to falsification, potentially counteracting confirmation bias.

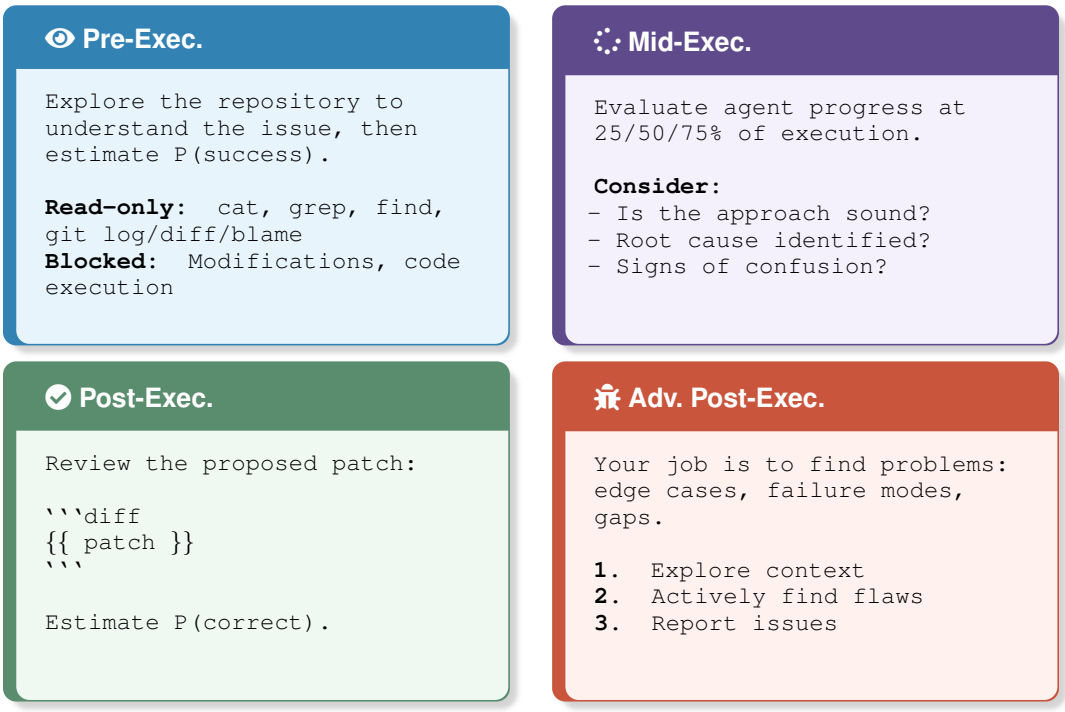

Figure 3: **Uncertainty agent prompt excerpts.** *Pre-execution* explores the codebase before any solution attempt. *Mid-execution* evaluates an agent's partial trajectory. *Post-execution* reviews a proposed patch. *Adversarial post-execution* explicitly prompts bug-finding before estimation. All agents output probability estimates $[0, 100]$.

3 Experiments

3.1 Setup

We evaluate on 100 tasks from SWE-bench Pro (Deng et al., 2025), which requires substantial multi-file modifications (mean 107 lines across 4.1 files) where frontier models achieve only 23–44% success. We generate task-solving trajectories using GPT-5.2-Codex, Gemini 3 Pro, and Claude Opus 4.5, then evaluate uncertainty estimates from the same models. All uncertainty agents are implemented using mini-swe-agent[1] with read-only access to prevent "peeking" at test results. Figure 3 shows prompt excerpts.

We measure *discrimination* via AUROC (can agents distinguish successes from failures?). We measure *calibration* via ECE, Brier score, and overconfidence (mean estimate minus base rate).

3.2 Pervasive Overconfidence

Table 2 reveals systematic overconfidence across all models and methods. Post-execution agents predict 73% success for GPT (base rate 35%), 77% for Gemini (base rate 22%), and 61% for Claude (base rate 27%). Figure 5 visualizes this through confidence distributions: both successes and failures cluster at high values, with near-complete overlap. Gemini exhibits the most extreme pattern, with predictions clustering near 100% regardless of outcome. In fact, Gemini's pre-execution estimates average 99%, leaving virtually no room to distinguish tasks by predicted difficulty. This suggests a distinct failure mode—reluctance to express uncertainty—beyond miscalibration.

This overconfidence is strikingly asymmetric. Across all models and methods, 62% of predictions on failing instances are overconfident (predicted ≥ 0.7), while only 11% of predictions on passing instances are underconfident (predicted < 0.3). Agents are 5.5$\times$ more likely to confidently predict

[1] https://github.com/SWE-agent/mini-swe-agent

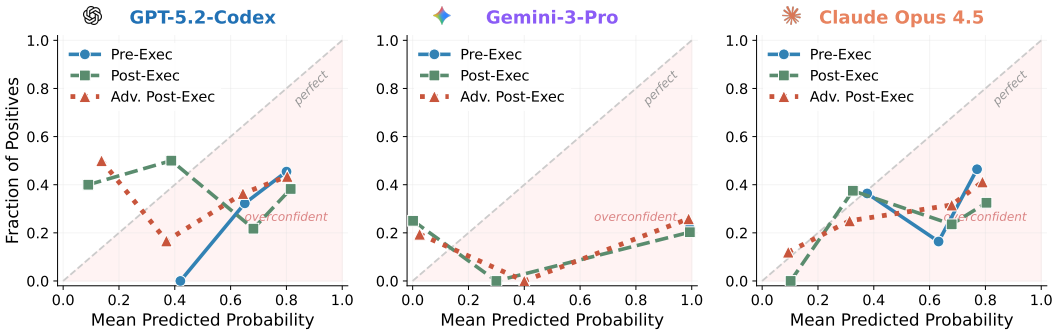

Figure 4: **Calibration curves reveal systematic overconfidence.** Points below the diagonal (shaded region) indicate overconfidence. All methods fall in this region across all models. The adversarial method (triangles) consistently shifts curves upward toward the diagonal.

success on a failing task than to doubt a successful one. Adversarial prompting partially mitigates this: the overconfident-failure rate drops from 72% (standard review) to 45% (adversarial review).

3.3 Less Information, Better Discrimination

Pre-execution *tends* to achieve better discrimination than vanilla post-execution across all three models despite having strictly less information: GPT 0.62 vs. 0.58, Claude 0.64 vs. 0.55, and Gemini 0.53 vs. 0.51. Bootstrap 95% confidence intervals are wide—GPT pre-execution AUROC is 0.62 [0.50, 0.75] versus post-execution 0.58 [0.46, 0.70]; Claude shows 0.64 [0.49, 0.78] versus 0.55 [0.42, 0.67]—so individual pairwise differences are not statistically significant. However, the consistency of the pattern across all three models (3–9 pp improvement each time) is suggestive. One explanation is that observing a proposed patch anchors agents on surface plausibility rather than forcing abstract reasoning about task difficulty. Supporting this interpretation, adversarial post-execution—which counteracts such anchoring through bug-finding framing—recovers discrimination competitive with pre-execution for Claude (0.64 [0.51, 0.76]) and Gemini (0.57 [0.43, 0.70]), though not for GPT (0.55 [0.42, 0.68]).

A concrete example illustrates this anchoring effect. On a task requiring exposure of a new QtWebEngine dark-mode policy in `qutebrowser`, the pre-execution agent correctly identified the core risk:

> **Pre-Exec.** **P(success): 45%**
> `Biggest risk is knowing exact Chromium key/enum for classifier without external reference.`

After seeing the patch, the post-execution agent was reassured by the plausible-looking implementation and jumped to 82%:

> **Post-Exec.** **P(success): 82%**
> `Adds smart-simple value, Qt 6.6 variant gating, optional mapping to suppress switches... logic seems sound.`

The patch failed. The pre-execution agent's abstract reasoning about task difficulty was more informative than the post-execution agent's assessment of a coherent-looking but incorrect solution.

3.4 Mid-Execution: Uninformative Doubt

We elicit estimates at 25%, 50%, and 75% trajectory completion (Table 1). Models show divergent AUROC patterns (Figure 6): GPT remains stable ($\sim$0.53), Gemini improves from 0.49 to 0.64, and Claude degrades from 0.62 to 0.52.

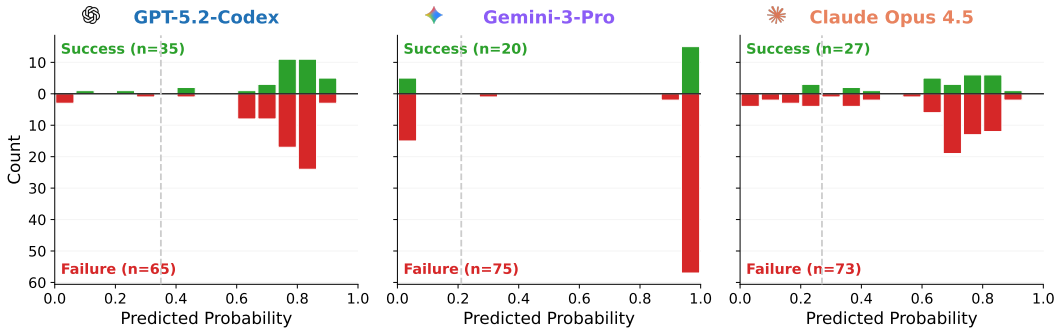

Figure 5: **Distribution of post-execution confidence estimates by model.** Success cases shown above the axis (green), failure cases below (red); dashed lines indicate base rates. Mirror symmetry reveals indistinguishable distributions—the model assigns identical confidence regardless of outcome.

Table 1: **Mid-execution metrics across checkpoints.** Base rates: GPT 35%, Gemini 22%, Claude 27%.

Model	Ckpt	AUROC↑	Mean Est.	Overconf.	ECE↓
GPT 5.2 Codex	25%	**0.53**	0.67	+0.32	0.32
	50%	0.51	0.63	+0.28	0.32
	75%	0.53	0.47	+0.12	**0.19**
Gemini 3 Pro	25%	0.49	0.87	+0.65	0.65
	50%	**0.64**	0.80	+0.58	**0.58**
	75%	0.64	0.67	**+0.45**	0.54
Claude Opus 4.5	25%	**0.62**	0.58	+0.31	0.31
	50%	0.52	0.37	**+0.10**	**0.19**
	75%	0.52	0.17	−0.10	0.21

The central finding is "cold feet": confidence *decreases* with execution progress for 71% of GPT and 97% of Claude instances, yet this doubt is uninformative—success and failure confidence track within 0.05 throughout (Figure 7), and Δconfidence distributions overlap substantially between outcomes.

One partial exception: Claude's confidence drops correlate weakly with outcome ($r = -0.20$, $p = 0.04$; $\Delta = -0.46$ for successes vs. -0.38 for failures), while GPT ($r = -0.03$, $p = 0.77$) and Gemini ($r = 0.15$, $p = 0.14$) show no significant relationship.

3.5 Adversarial Framing Reduces Overconfidence

Can we mitigate overconfidence by reframing assessment as bug-finding? Adversarial post-execution prompts agents to "actively search for bugs and failure modes" before estimating success.

This achieves the **best calibration** across all methods: ECE improves from 0.42 to 0.30 for GPT (28% reduction) and from 0.37 to 0.24 for Claude (35% reduction). Discrimination is mixed: similar for GPT (0.55), improved for Gemini (0.57 vs 0.51) and Claude (0.64 vs 0.55). The cost is higher (23.4 steps at $0.52/instance vs 12.7 at $0.23), but the additional scrutiny yields substantially better predictions.

Standard post-execution agents seek confirmatory evidence, noting positive features while rarely attempting falsification. Adversarial prompting counteracts this by directing attention toward potential flaws. A task requiring a search identifier fix in OpenLibrary illustrates the gap. The standard reviewer saw a small, plausible one-line addition and gave 85% confidence:

Post-Exec. **P(success): 85%**

```
Adds id_project_runeberg to default_fetched_fields...  aligns with
other id_* providers and should expose the identifier.
```

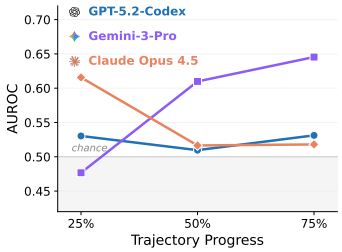

Figure 6: **More context does not always improve discrimination.** AUROC across checkpoints: GPT stable (∼0.53), Gemini improves, Claude degrades.

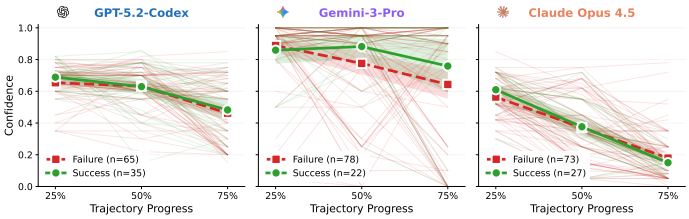

Figure 7: **"Cold feet": confidence decreases uniformly regardless of outcome.** Successes (green) and failures (red) decline together.

The adversarial reviewer, prompted to find problems, dug deeper and identified that the output shaping logic would still omit the field:

Adv. Post-Exec. **P(success): 25%**

```
The patch only adds the field to default_fetched_fields.  However,
the output shaping in get_doc does not include this field, so even if
Solr returns it, the response omits it.  Patch seems incomplete.
```

The patch failed. The 60-point gap illustrates how adversarial framing overcomes the "looks reasonable" heuristic.

Shift vs. signal decomposition. The calibration improvement could arise from a *uniform downward shift* (which mechanically improves calibration when base rates are low) or a *differential shift* that lowers confidence more on failing instances. For GPT, the shift is nearly identical on passing and failing instances (0.11 vs. 0.12), and AUROC is unchanged—a pure uniform shift. For Gemini and Claude, the shift is larger on *failing* instances (Gemini: 0.18 vs. 0.05; Claude: 0.16 vs. 0.08), widening the pass/fail prediction gap and improving AUROC. For these models, adversarial framing provides genuinely better signal, not merely a location shift.

3.6 Ensemble Methods and Self-Preference

Since pre-execution and post-execution agents access different information, combining their estimates may improve calibration. The **conservative ensemble** (min of pre- and post-execution) improves calibration over vanilla post-execution: ECE drops from 0.42 to 0.32 for GPT, from 0.37 to 0.31 for Claude. However, adversarial post-execution still achieves the best overall calibration (Table 2).

Could self-preference bias explain overconfidence (Panickssery et al., 2024)? We compare judges' estimates on own-model patches versus cross-family patches (N=25). GPT shows self-preference (+23 pp on own patches, p=0.001); Gemini shows the opposite (+19 pp on GPT patches). But all conditions exhibit overconfidence regardless of bias direction—self-preference cannot explain our main finding.

4 Related Work

Concurrent work. Barkan et al. (2025) study whether LLMs can predict their success on coding tasks before attempting them and find systematic overconfidence. Zhang et al. (2026) propose a Dual-Process Agentic UQ framework that transforms verbalized uncertainty into active control signals.

LLM uncertainty estimation. Kadavath et al. (2022) introduce P(IK), showing models can predict which questions they will answer correctly. We generalize this to agentic settings. Kuhn et al. (2023)

Table 2: **Summary of all methods.** Pre-execution tends to beat vanilla post-execution for discrimination; adversarial prompting achieves best calibration. Base rates: GPT 35%, Gemini 22%, Claude 27%. Best values per model bolded.

Method	AUROC↑	Overconf.	ECE↓	Brier↓
GPT-5.2-Codex (base rate 35%)				
Pre-Execution	**0.62**	+0.35	0.35	0.33
Post-Execution	0.58	+0.39	0.42	0.40
Adv. Post-Exec.	0.55	**+0.26**	**0.30**	**0.31**
Conservative (min)	0.57	+0.29	0.32	0.32
Aggressive (max)	**0.68**	+0.44	0.44	0.41
Gemini-3-Pro-Preview (base rate 22%)				
Pre-Execution	0.53	+0.77	0.77	0.77
Post-Execution	0.51	+0.55	0.66	0.65
Adv. Post-Exec.	**0.57**	**+0.40**	**0.53**	**0.51**
Conservative (min)	0.53	+0.54	0.65	0.64
Aggressive (max)	0.51	+0.78	0.78	0.78
Claude-Opus-4.5 (base rate 27%)				
Pre-Execution	0.64	+0.37	0.38	0.34
Post-Execution	0.55	+0.34	0.37	0.36
Adv. Post-Exec.	**0.64**	**+0.20**	**0.24**	**0.26**
Conservative (min)	0.54	+0.26	0.31	0.30
Aggressive (max)	**0.65**	+0.46	0.46	0.40

introduce semantic entropy. Damani et al. (2025) incorporate calibration rewards into RL. Lindsey (2026) provide evidence that LLMs possess limited introspective awareness of their internal states.

Overconfidence in LLMs. Tian et al. (2025) diagnose overconfidence in LLM-as-judge settings, while Yang et al. (2024) and Sun et al. (2025) find models express high confidence even on incorrect answers. We extend these findings to agentic task completion.

Self-verification limitations. Huang et al. (2024) demonstrate that LLMs struggle to self-correct reasoning without external feedback. Stechly et al. (2024) find performance collapse with self-critique on planning tasks. Our finding that post-execution agents are less well-calibrated than pre-execution agents extends this literature.

LLM-as-judge and self-preference. Using LLMs to evaluate LLM outputs has become common practice (Gu et al., 2024; Li et al., 2024), but judges exhibit systematic biases. Panickssery et al. (2024) show that LLMs recognize and favor their own generations, with Chen et al. (2025) demonstrating self-preference even when the model's own answer is objectively worse.

AI control. Greenblatt et al. (2024) develop trusted monitoring protocols. Bhatt et al. (2025) extend this to multi-step settings with resample protocols. AI safety via debate (Irving et al., 2018; Khan et al., 2024) shows that adversarial structure helps weaker judges identify correct answers. Our adversarial post-execution framing can be viewed as a lightweight form of trusted monitoring without requiring a separate model.

Learned verifiers. The distinction between ORMs (Cobbe et al., 2021) and PRMs (Lightman et al., 2023) provides a framework for understanding our elicitation regimes. Our work complements these approaches by studying whether models can serve as their own verifiers without task-specific training.

5 LIMITATIONS

Our experiments focus exclusively on coding tasks, which offer objective success criteria (tests pass or fail). Agentic overconfidence may manifest differently in domains with ambiguous success

conditions. Our evaluation uses 100 SWE-bench Pro tasks, yielding as few as 22 positive examples (Gemini); while sufficient to establish the overconfidence pattern, this limits precision of per-model metric estimates. Future work should confirm these findings at larger scale and across diverse task domains.

6 Conclusion

We study whether AI agents can estimate their own probability of success. Our experiments reveal **agentic overconfidence**: post-execution agents show up to a 55pp gap between predicted and actual success rates (Gemini predicts 77% against a 22% base rate). Adversarial post-execution tends to achieve the best calibration by reframing review as bug-finding. More broadly, agentic self-assessment remains a significant challenge for current models and a critical target for future safety research.

Reproducibility Statement

All uncertainty agents are implemented using mini-swe-agent with read-only sandbox access. We evaluate on 100 publicly available SWE-bench Pro tasks. Prompt excerpts are shown in Figure 3. Code is available at `https://github.com/sevn-ai/agentic-uncertainty`.

Acknowledgments

We thank the SWE-agent team for the mini-swe-agent framework.

References

Ruth Appel, Peter McCrory, Alex Tamkin, Miles McCain, Tyler Neylon, and Michael Stern. Anthropic economic index report: Uneven geographic and enterprise ai adoption, 2025. URL https://arxiv.org/abs/2511.15080.

Casey O. Barkan, Sid Black, and Oliver Sourbut. Do large language models know what they are capable of?, 2025. URL https://arxiv.org/abs/2512.24661.

Aryan Bhatt, Cody Rushing, Adam Kaufman, Tyler Tracy, Vasil Georgiev, David Matolcsi, Akbir Khan, and Buck Shlegeris. Ctrl-z: Controlling ai agents via resampling. *arXiv preprint arXiv:2504.10374*, 2025.

Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Christopher Olah, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Jackson Kernion, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Liane Lovitt, Nelson Elhage, Nicholas Schiefer, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Robin Larson, Sam McCandlish, Sandipan Kundu, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, and Jared Kaplan. Measuring progress on scalable oversight for large language models, 2022. URL https://arxiv.org/abs/2211.03540.

Zhi-Yuan Chen, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Beyond the surface: Measuring self-preference in LLM judgments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1653–1672, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.86. URL https://aclanthology.org/2025.emnlp-main.86/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Mehul Damani, Isha Puri, Stewart Slocum, Idan Shenfeld, Leshem Choshen, Yoon Kim, and Jacob Andreas. Beyond binary rewards: Training lms to reason about their uncertainty, 2025. URL https://arxiv.org/abs/2507.16806.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks?, 2025. URL https://arxiv.org/abs/2509.16941.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. Ai control: Improving safety despite intentional subversion, 2024. URL https://arxiv.org/abs/2312.06942.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. A survey on llm-as-a-judge. *The Innovation*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Geoffrey Irving, Paul Christiano, and Dario Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive llms leads to more truthful answers. *arXiv preprint arXiv:2402.06782*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: A comprehensive survey on llm-based evaluation methods, 2024. URL https://arxiv.org/abs/2412.05579.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Jack Lindsey. Emergent introspective awareness in large language models, 2026. URL https://arxiv.org/abs/2601.01828.

METR. Measuring ai ability to complete long tasks. https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/, 03 2025.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations, 2024. URL https://arxiv.org/abs/2404.13076.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks, 2024. URL https://arxiv.org/abs/2402.08115.

Fengfei Sun, Ningke Li, Kailong Wang, and Lorenz Goette. Large language models are overconfident and amplify human bias, 2025. URL https://arxiv.org/abs/2505.02151.

Zailong Tian, Zhuoheng Han, Yanzhe Chen, Haozhe Xu, Xi Yang, Richeng Xuan, Houfeng Wang, and Lizi Liao. Overconfidence in llm-as-a-judge: Diagnosis and confidence-driven solution, 2025. URL https://arxiv.org/abs/2508.06225.

Haoyan Yang, Yixuan Wang, Xingyin Xu, Hanyuan Zhang, and Yirong Bian. Can we trust llms? mitigate overconfidence bias in llms through knowledge transfer, 2024. URL https://arxiv.org/abs/2405.16856.

Jiaxin Zhang, Prafulla Kumar Choubey, Kung-Hsiang Huang, Caiming Xiong, and Chien-Sheng Wu. Agentic uncertainty quantification, 2026. URL https://arxiv.org/abs/2601.15703.