UNCERTAINTY-AWARE SELF-CORRECTION FOR CODING AGENTS

Jason Almeida, Lokesh Sai Dasari, Anubhav Pal, Tinuade Adeleke,*
Sean Wu, Ruizhe Li
Algoverse AI Research
Palo Alto, CA, USA

ABSTRACT

Recent advances in large language models (LLMs) have enabled agentic systems that perform complex, multi-step tasks in realistic environments, particularly in software engineering settings where agents must navigate code bases, plan actions, execute code, and iteratively adapt based on environmental feedback. Despite their capabilities, agent reliability remains a critical challenge: errors made early in an agent's trajectory can propagate, leading to incorrect patches, wasted computation, or misleading confidence. Most existing uncertainty estimation methods focus on single-turn outputs and do not account for uncertainty accumulation across multi-step reasoning.

In this work, we adapt and extend Situational Awareness Uncertainty Propagation (SAUP) to coding agents operating on SWE-Rebench. We propose a simplified variant of SAUP that replaces learned semantic distance metrics with API-derived signals and heuristic action weights, making it practical for black-box API settings. We demonstrate how step level uncertainty estimation can be propagated across an agent's trajectory and used to trigger self-correction when confidence is low.

By intervening selectively at high-uncertainty steps, our approach improves final task success while avoiding unnecessary computation. Across three frontier models (GPT-5, Claude Opus 4.5, and DeepSeek V3.2), uncertainty-aware resampling reduces mean trajectory uncertainty by 6–20% relative and improves pass@1 by up to 15.6 absolute percentage points, with latency overhead of 1.2–3.0× depending on the model.

1 INTRODUCTION

Recent advances in large language models (LLMs) have enabled the emergence of agentic systems that go far beyond single-turn prompting. Modern LLM-based agents are capable of performing complex, multi-step tasks in realistic environments, particularly in software engineering settings where agents must understand large code bases, plan sequences of actions, execute code, interpret test results, and iteratively adapt their behavior based on feedback from the environment (Jimenez et al., 2024; Liu et al., 2023). Coding agents operating in these settings increasingly resemble human developers, interacting with development tools and execution environments over many steps rather than producing isolated code snippets.

Despite their impressive capabilities, the reliability of such agents remains a critical challenge. Errors made early in an agent's reasoning or action sequence can propagate through subsequent steps, leading to incorrect patches, wasted computation, or misleading confidence in faulty solutions. As tasks grow longer and more complex, understanding and managing uncertainty becomes essential for improving agent reliability Xia et al. (2025); Huang et al. (2025). However, most existing uncertainty estimation methods for LLMs focus exclusively on the model's final output and do not account for uncertainty accumulation across multiple reasoning and action steps.

*Corresponding author. Please direct correspondence to: tinuade@algoverseairesearch.org

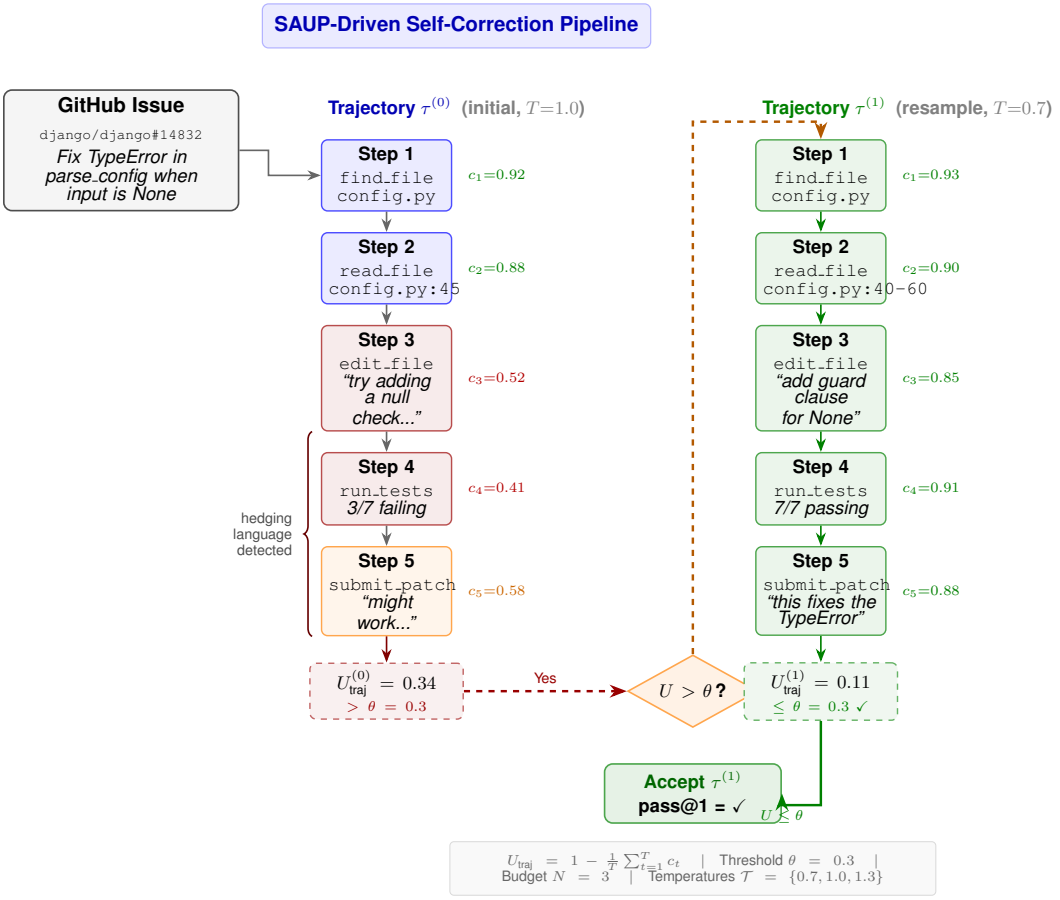

Figure 1: **Overview of SAUP-driven self-correction on a SWE-Rebench task.** Given a GitHub issue (`django/django#14832`), the agent generates an initial trajectory $\tau^{(0)}$ at temperature T=1.0. At each step, we extract a confidence score c_t using token probabilities, API signals, and semantic markers. In $\tau^{(0)}$, the agent produces a tentative edit with hedging language (Step 3: *"try adding a null check"*), leading to test failures (Step 4) and high trajectory uncertainty $U_{\text{traj}}^{(0)} = 0.34 > \theta$. This triggers resampling: a new trajectory $\tau^{(1)}$ is generated at T=0.7, where the agent produces a more decisive fix with confident language and all tests passing, yielding $U_{\text{traj}}^{(1)} = 0.11 \leq \theta$. The lower-uncertainty trajectory is accepted.

Moreover, uncertainty estimation techniques have largely been explored in the context of code generation rather than full agentic workflows that involve interaction with external environments (He et al., 2025; Zhu et al., 2025). Coding agents present a particularly challenging setting: they must not only generate code, but also interpret execution outcomes, revise plans, and decide when corrective action is needed. Accurately estimating uncertainty at each step of this process offers an opportunity to prevent error propagation and enable targeted self-correction before failures compound.

In this work, we make the following contributions:

1. We adapt Situational Awareness Uncertainty Propagation (SAUP) (Zhao et al., 2025) for coding agents in realistic SWE scenarios, replacing its learned HMM-based situational weights with heuristic action type weights and substituting semantic embedding distances with API level confidence signals.

2. We propose a threshold based adaptive resampling policy that triggers self-correction only when trajectory-level uncertainty exceeds a learned threshold, balancing reliability gains against compute cost.

3. We evaluate our approach on SWE-Rebench (Badertdinov et al., 2025) across three frontier models: GPT-5 (Singh et al., 2025), Claude Opus 4.5 (Anthropic, 2025), and DeepSeek V3.2 (DeepSeek-AI et al., 2025) demonstrating that SAUP-driven resampling consistently improves pass@1 while keeping latency overhead modest.

2 Related Work

LLM-Based Agent Evaluation. AgentBench (Liu et al., 2023) is a general evaluation framework for LLM-based agents operating in interactive, multi-step environments spanning operating systems, databases, knowledge graphs, web shopping, and embodied tasks. Our work builds on the AgentBench framework but focuses specifically on uncertainty-driven self-correction in software engineering tasks.

SWE-bench and SWE-Rebench. SWE-bench (Jimenez et al., 2024) evaluates LLMs on real-world GitHub issues, requiring models to generate patches verified by executing repository test suites in a Dockerized environment. SWE-Rebench (Badertdinov et al., 2025) extends this setup by emphasizing reproducibility, standardized execution, and agent style interaction. We adopt SWE-Rebench as our primary evaluation setting because it reflects the full complexity of coding agents in practice: long-horizon reasoning, environmental feedback, and strict correctness criteria.

Uncertainty Estimation for LLMs. Prior work on uncertainty estimation includes token-level entropy (Malinin & Gales, 2021), mean token entropy, verbalized confidence scores (Kadavath et al., 2022; Tian et al., 2023), and probability of truth estimation (Kadavath et al., 2022). While useful, these methods are typically applied to single outputs and do not capture how uncertainty evolves across multi-step reasoning or interaction.

Uncertainty in Code Generation. Several recent works explore uncertainty-aware code generation, including Uncert-CoT (Zhu et al., 2025) and AdaDec (He et al., 2025), which use uncertainty signals to guide decoding or reranking. While effective in prompt based generation, these methods operate at the level of a single generation step and do not address uncertainty propagation in agents interacting with external environments.

SAUP. Situational Awareness Uncertainty Propagation (Zhao et al., 2025) introduces a framework for estimating and propagating uncertainty across the steps of an LLM agent's trajectory while performing natural language tasks. The original SAUP computes step level uncertainty using normalized predictive entropy, derives situational weights from two learned semantic distance metrics; Inquiry Drift (measuring trajectory divergence from the original question) and Inference Gap (measuring actionobservation mismatch) and aggregates them via a Continuous Hidden Markov Model (CHMM) trained with the Baum-Welch algorithm. In this work, we adapt SAUP for black-box API settings where logprobs and embedding access are limited: we replace semantic distance metrics with API-derived signals (finish reason, action type heuristics, hedging phrase detection), and substitute the CHMM with fixed action criticality weights, and use arithmetic mean aggregation rather than RMS.

SWE-PRM. SWE-PRM (Gandhi et al., 2025) introduces a process reward model for software engineering agents that scores partial trajectories and enables resuming execution from sub-optimal intermediate states. While SWE-PRM focuses on *step-level reward modeling* to identify and branch from weak trajectory stages, our approach operates at the *full-trajectory level*: we resample entire trajectories when aggregated uncertainty exceeds a threshold, and select the candidate with lowest uncertainty. The two strategies are complementary, SWE-PRM's fine-grained branching could be combined with our trajectory-level resampling to enable both local recovery and global selection.

3 METHODS

3.1 FRAMEWORK

Figure 1 illustrates our approach on a representative SWE-Rebench task: an agent attempts to resolve a GitHub issue, and SAUP monitors confidence at each step, triggering resampling when trajectory uncertainty exceeds the threshold. We extend AgentBench (Liu et al., 2023) with hierarchical uncertainty quantification. Agents interact with task environments through multi-step reasoning: at each step t, the agent observes state o_t, selects an action a_t, and receives feedback from the environment. Our framework quantifies uncertainty at each step and aggregates it into a trajectory-level measure that informs adaptive resampling decisions.

3.2 UNCERTAINTY QUANTIFICATION

We employ a multi-signal uncertainty estimation approach combining token-level, action-level, and trajectory-level confidence measures. At each agent step t, we extract a confidence score $c_t \in [0, 1]$ using a hierarchical strategy with multiple fallback sources.

Token-level confidence. When the model API returns log probabilities, we compute step confidence as the geometric mean of token probabilities:

$$c_t = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log p(x_i \mid x_{<i})\right) \tag{1}$$

where $p(x_i \mid x_{<i})$ is the probability of token x_i given the preceding context.

API signal extraction. For function-calling APIs where log probabilities are unavailable (e.g., OpenAI tool use), we derive baseline confidence from response metadata by mapping the `finish_reason` field: `stop` $\rightarrow$ 0.85, `tool_calls` $\rightarrow$ 0.80, `length` $\rightarrow$ 0.50, and `content_filter` $\rightarrow$ 0.30.

The intuition behind this mapping is that `finish_reason` provides a coarse but informative signal about the model's generation process. A `stop` termination indicates the model reached a natural completion point and it generated the full response it intended, which we treat as a mild positive signal (0.85). A `tool_calls` termination indicates the model elected to invoke a tool rather than produce a final answer; this is expected behavior in agentic settings but reflects an intermediate state where the model is still gathering information, warranting slightly lower baseline confidence (0.80). A `length` termination, by contrast, means the model's response was truncated by the context window limit before it could finish, a strong indicator that the model was unable to fully express its reasoning or that the task induced an unusually long output, both of which correlate with lower reliability (0.50). Finally, `content_filter` indicates the model's output was blocked or truncated by a safety filter, which we treat as the lowest-confidence signal (0.30) since the intended output was suppressed entirely. We emphasize that these values serve as *priors* that are subsequently modulated by the action complexity and semantic signals described below; they are not intended as calibrated probabilities but rather as an ordinal ranking of how complete and intentional each step's output was.

Action complexity analysis. For tool-calling agents, we adjust confidence based on the action type, reflecting the observation that different actions carry fundamentally different risk profiles in software engineering workflows. Read-only operations (`read_file`, `list_dir`) receive higher base confidence (0.90) because they are inherently safe, they cannot alter the repository state and their failure modes are limited to reading the wrong file, which is easily recovered from. High-stakes write operations (`edit_file`, `submit_patch`) receive lower base confidence (0.75) because they modify code and can introduce bugs, and because the model must simultaneously reason about code semantics, patch correctness, and integration effects.

Within bash command calls, we further analyze command patterns. Destructive patterns (`rm -rf`, `sudo`, `chmod`) reduce confidence by up to 0.15, reflecting both the irreversibility of such operations

and the observation that well functioning agents rarely need them for issue resolution; their appearance often signals the agent has entered an unusual execution path. Conversely, safe diagnostic patterns (`ls`, `git status`, `cat`, `grep`) increase confidence by up to 0.05, a deliberately asymmetric adjustment, since safe actions provide weak positive evidence (the agent is exploring rationally) but destructive actions provide strong negative evidence (the agent may be flailing). This asymmetry is motivated by the empirical observation that failed trajectories disproportionately contain risky commands, while successful trajectories contain a mix of both safe and moderately complex operations.

Semantic uncertainty markers. A key insight motivating our approach is that LLMs often display their own uncertainty through natural language cues in their reasoning traces, even when the underlying token probabilities are unavailable. We exploit this by applying pattern based analysis to detect hedging language in both reasoning content and tool call arguments. Hedging phrases (e.g., "I think," "probably," "might be," "let me try," "assume") reduce confidence, while confident phrases (e.g., "this fixes," "definitely," "will work," "the issue is") increase it:

$$\Delta_s = \min(0.02n_c,\ 0.10) - \min(0.03n_h,\ 0.15) \tag{2}$$

where n_c and n_h are the counts of confident and hedging phrases, respectively. We emphasize that the coefficients in Equation 2 are not tuned to optimize benchmark performance, but are intended to encode qualitative asymmetry between linguistic signals. In practice, we observed that moderate variation in these values produced similar behavior across runs, indicating that the uncertainty signal is robust to reasonable parameter changes. These coefficients should be interpreted as heuristic weighting factors rather than calibrated probabilities.

Several design choices in Equation 2 merit discussion. First, the hedging coefficient (0.03) is $1.5\times$ larger than the confidence coefficient (0.02), reflecting the asymmetric informativeness of linguistic cues: when an LLM explicitly hedges, this is a strong signal of genuine uncertainty, whereas confident language may simply reflect the model's default assertive style and is less reliably predictive of correctness. Second, both terms are capped (at 0.15 and 0.10, respectively) to prevent any single signal source from dominating the overall confidence score; without capping, a verbose reasoning trace full of hedging phrases could drive confidence to near zero regardless of the token-level or action-level signals. Third, we apply semantic analysis to *tool call arguments* in addition to reasoning content, because in practice agents often embed uncertainty cues in the strings they pass to tools for example, an edit command with the description "try this approach" is meaningfully less reliable than one stating "fix the null pointer dereference." This signal is particularly valuable for Claude Opus 4.5, where `finish_reason` is nearly constant across runs (see Table 1), making semantic markers the primary source of within model confidence variation.

Trajectory-level aggregation. We aggregate step confidences into trajectory uncertainty:

$$U_{\text{traj}} = 1 - \frac{1}{T}\sum_{t=1}^{T} c_t \tag{3}$$

Note that this differs from the original SAUP aggregation, which uses weighted Root Mean Square (RMS):

$$U_{\text{agent}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(W_i U_i)^2} \tag{4}$$

We adopt arithmetic mean for computational simplicity and because our heuristic weights lack the learned calibration that would justify RMS aggregation.

We additionally track minimum confidence $c_{\min} = \min_t c_t$, confidence trend (comparing first half and second half means), and the count of high uncertainty steps $|\{t : c_t < 0.5\}|$.

3.3 Threshold-Based Resampling

Beyond SAUP's uncertainty quantification, we introduce adaptive threshold based resampling to improve reliability. Algorithm 1 summarizes the procedure. As a control, we also evaluate a random resampling baseline that matches the SAUP policy's resampling budget but removes the uncertainty trigger. Resampling is activated with a probability equal to each model's observed SAUP resampling

Algorithm 1 SAUP-Driven Adaptive Resampling

Require: Task instance, threshold θ, budget N, temperatures $\mathcal{T} = \{0.7, 1.0, 1.3\}$
Generate initial trajectory $\tau^{(0)}$ at $T = 1.0$
Compute $U^{(0)}_{\text{agent}}$ via Eq. 3
if $U^{(0)}_{\text{agent}} \leq \theta$ **then**
  return $\tau^{(0)}$ {Accept first-pass trajectory}
end if
for $r = 1, \ldots, N$ **do**
  Sample $T_r \sim \text{Uniform}(\mathcal{T})$
  Generate trajectory $\tau^{(r)}$ at temperature T_r
  Compute $U^{(r)}_{\text{agent}}$
  if $U^{(r)}_{\text{agent}} \leq \theta$ **then**
    return $\tau^{(r)}$ {Early stopping}
  end if
end for
return $\arg\min_{r \in \{0,\ldots,N\}} U^{(r)}_{\text{agent}}$ {Select lowest uncertainty}

rate. The final trajectory is selected using the same minimum-uncertainty rule. This baseline isolates the value of uncertainty-guided triggering.

3.4 Grid Search for Hyperparameters

We perform a grid search over the uncertainty threshold $\theta \in \{0.3, 0.5, 0.7, 0.9\}$ and the resampling budget $N \in \{1, 3, 5, 8\}$ on a held-out validation subset. The full grid search (Table 5 in the Appendix) indicates that $\theta = 0.3$ and $N = 8$ are globally optimal. However, to reduce computation, we fix $N = 3$ for the main experiments; under this budget, the optimal threshold remains $\theta = 0.3$.

3.5 Evaluation Metrics

We evaluate both task performance and uncertainty calibration.

Task performance. **Pass@1**: Success rate on the first generated trajectory (with SAUP-driven resampling when applicable). **Baseline pass@1**: Success rate without any self-correction (single trajectory, no resampling).

Uncertainty calibration. **Spearman correlation** (ρ): rank correlation between step confidence c_t and binary task success. **ECE**: expected calibration error, the weighted average of $|\text{accuracy} - \text{confidence}|$ across 10 bins. **Brier Score**: mean squared error between confidence and outcome, $\frac{1}{N}\sum_i (c_i - y_i)^2$. **AUROC**: area under the ROC curve, measuring whether successful trajectories receive higher confidence than failed ones.

Latency. Latency multiplier: ratio $t_{\text{SAUP}}/t_{\text{baseline}}$, measuring additional cost from resampling.

3.6 Datasets and Implementation

We evaluate on SWE-Rebench (Badertdinov et al., 2025), consisting of 500 GitHub issue resolution tasks with pre-built docker images for reproducible environment grounded evaluation. Experiments use GPT-5 (Singh et al., 2025), Claude Opus 4.5 (Anthropic, 2025), and DeepSeek V3.2 (DeepSeek-AI et al., 2025) with Docker Compose infrastructure, YAML-configured tasks, and resumable runs. All models are accessed via their respective APIs; confidence signals are derived from logprobs where available and from `finish_reason` metadata otherwise.

All API-based experiments were conducted between December 2025 and March 2026. We used the following model versions available at that time: GPT-5 via the OpenAI Chat Completions API (`gpt-5.1-2025-04-14`), Claude Opus 4.5 via the Anthropic Messages

Table 1: SAUP uncertainty calibration on SWE-Rebench. ↑: higher is better; ↓: lower is better. Best results per metric are **bolded**. †Degenerate due to constant confidence scores.

Metric	GPT-5	Claude Opus 4.5	DeepSeek V3.2
Spearman ρ (↑)	**0.400**	$1.000^{\dagger}$	0.109
ECE (↓)	0.148	0.300	**0.101**
Brier Score (↓)	**0.022**	0.090	0.104
AUROC (↑)	0.500	0.500	**0.507**

API (`claude-opus-4-5-20250514`), and DeepSeek V3.2 via the DeepSeek Chat API (`deepseek-chat`). All models were accessed through their respective commercial APIs with default rate limits. Token-level logprobs were available only for GPT-5; for Claude and DeepSeek, confidence was derived from `finish_reason` and semantic signals as described in Section 3.2. Model behavior may differ under subsequent API updates or version changes. All experiments were conducted using publicly available commercial APIs under standard academic usage conditions, with no privileged access, fine-tuning, or internal model modifications.

4 RESULTS

4.1 SAUP UNCERTAINTY CALIBRATION

Table 1 presents the uncertainty calibration metrics across all models on SWE-Rebench. All models exhibit AUROC scores near 0.5, indicating that confidence scores derived from API signals do not reliably discriminate between successful and failed trajectories. DeepSeek V3.2 achieves the lowest ECE (0.101), suggesting better calibration between stated confidence and actual success rates, but its weak Spearman correlation ($\rho = 0.109$) reveals that even well calibrated models struggle to produce confidence scores that correlate with outcomes. GPT-5 shows stronger correlation ($\rho = 0.400$) but higher ECE (0.148), suggesting a trade-off between discrimination and calibration. Claude Opus 4.5 produces near constant confidence scores across runs due to its reliance on `tool_calls` as the dominant finish reason, resulting in a degenerate correlation ($\rho = 1.000^{\dagger}$). This highlights a fundamental limitation of relying solely on API-level finish-reason signals for uncertainty estimation.

Despite these calibration limitations, the propagated trajectory-level uncertainty U_{traj} proves useful as a *relative* signal for triggering resampling, as we show in the following sections.

Why resampling helps despite weak step-level discrimination. The near-random AUROC (≈ 0.5) indicates that step level confidence scores cannot reliably distinguish successful from failed trajectories on a per instance basis. At first glance, this appears to conflict with the consistent pass@1 improvements observed under uncertainty-aware resampling. The resolution lies in the difference between *pointwise discrimination* and *distributional selection*. AUROC measures whether a single confidence score can rank one trajectory above another, which requires fine grained per instance calibration. Resampling, by contrast, operates at the *population level*: it identifies a subset of trajectories whose aggregated uncertainty exceeds a threshold, regenerates them, and selects the candidate with the lowest uncertainty among k alternatives. Even when individual confidence scores are weakly informative, regenerating high-uncertainty trajectories with diverse temperatures introduces variation that increases the probability of sampling a correct solution. The minimum uncertainty selector then acts as a soft filter, preferring trajectories that are at least internally more consistent. In effect, resampling exploits the variance of the generation process rather than the precision of the confidence estimator—the threshold serves as a coarse "retry trigger" rather than a fine-grained classifier, and the gains come from giving the model additional chances on its most uncertain attempts. Our results suggest that API-level signals alone are insufficient for reliable success/failure discrimination, although they remain useful as relative triggers for resampling.

Table 2: SAUP-driven self-correction performance on SWE-Rebench ($\theta = 0.3$, $N = 3$). Latency is reported as a multiplier relative to single-pass baseline runtime.

	GPT-5		Claude Opus 4.5		DeepSeek V3.2	
Method	pass@1	Latency	pass@1	Latency	pass@1	Latency
Baseline (no regen)	0.600	1.0×	0.660	1.0×	0.580	1.0×
Random Resampling (matched p)	0.688	3.0×	0.714	1.2×	0.664	2.8×
SAUP Self-Correction	**0.756**	3.0×	**0.780**	1.2×	**0.739**	2.8×
Δ Improvement	+0.156	—	+0.120	—	+0.159	—

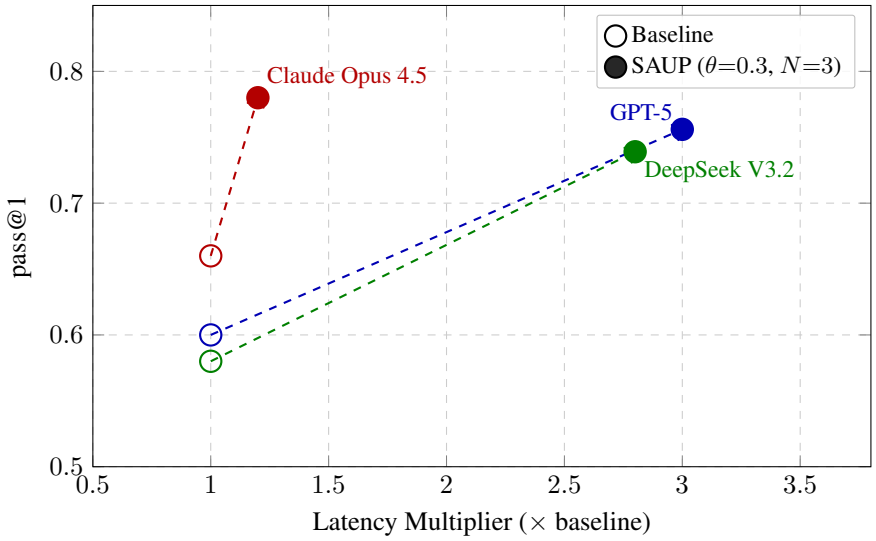

Figure 2: Pass@1 vs. relative latency for SAUP-driven self-correction on SWE-Rebench. Each arrow connects a model's baseline performance (open circle) to its SAUP-corrected performance (filled circle). Claude Opus 4.5 achieves the best latency–accuracy trade-off, while DeepSeek V3.2 and GPT-5 show larger absolute gains at higher latency cost.

4.2 SAUP-Driven Self-Correction

4.2.1 Main Results

Table 2 compares baseline performance (single trajectory, no resampling) against SAUP-driven self-correction with the optimal main-experiment configuration ($\theta = 0.3$, $N = 3$). All three models show meaningful improvements in pass@1 with SAUP-driven resampling. GPT-5 improves from a baseline of 0.600 to 0.756 (+15.6 pp), DeepSeek V3.2 from 0.580 to 0.739 (+15.9 pp), and Claude Opus 4.5 from 0.660 to 0.780 (+12.0 pp). These gains come at modest latency overhead: Claude adds only 1.2× over baseline, GPT-5 incurs 3.0×, and DeepSeek, which triggers resampling most frequently, incurs 2.8×. Figure 2 visualizes the pass@1 improvement against latency cost per model.

To determine whether the gains from SAUP arise from the uncertainty signal rather than from additional attempts we compare against a random resampling baseline with the same budget ($N = 3$) and matched expected resampling frequency. Random resampling improves over the single-pass baseline but remains below SAUP across all three models. The uncertainty signal helps to identify which trajectories should be retried because it offers more value than just providing additional sample allocation.

Table 3: Optimal (θ, N) configuration per model on SWE-Rebench. Pass@1 and latency under the best grid-search configuration.

Model	**Optimal θ**	**Optimal N**	**pass@1**	**Latency ($\times$)**
GPT-5	0.3	8	**0.955**	4.97$\times$
Claude Opus 4.5	0.3	3	0.780	1.2$\times$
DeepSeek V3.2	0.3	3	0.739	2.8$\times$

Table 4: Resampling dynamics under SAUP self-correction ($\theta = 0.3$, $N = 3$) on SWE-Rebench (500 tasks per model).

Metric	**GPT-5**	**Claude Opus 4.5**	**DeepSeek V3.2**
Tasks resampled	101 (20.2%)	76 (15.2%)	139 (27.9%)
Mean base uncertainty	0.237	0.230	0.289
Mean final uncertainty	0.213	0.211	0.230
Δ uncertainty	-0.024	-0.019	-0.059
Mean latency (s)	37.6	18.3	84.6

4.2.2 Joint Tuning of (θ, N)

Table 3 reports the optimal (θ, N) configuration per model identified via grid search. GPT-5 achieves 0.955 pass@1 at its optimal configuration ($\theta = 0.3$, $N = 8$), but at a substantial $4.97\times$ latency cost. This demonstrates that with a larger resampling budget, SAUP can push pass@1 well above the $N = 3$ operating point, though diminishing returns set in rapidly. The full grid search results are reported in Appendix A.

4.2.3 Resampling Dynamics

Table 4 summarizes the resampling behavior across models. Claude Opus 4.5 triggers resampling least frequently (15.2% of tasks), consistent with its lower baseline uncertainty and more stable trajectories. GPT-5 resamples at an intermediate rate (20.2%), while DeepSeek V3.2 resamples most often (27.9%), reflecting its higher baseline uncertainty.

All models benefit from resampling. DeepSeek shows the largest absolute reduction in mean uncertainty ($0.289 \to 0.230$, $\Delta = 0.059$), GPT-5 a moderate reduction ($0.237 \to 0.213$, $\Delta = 0.024$), and Claude the smallest ($0.230 \to 0.211$, $\Delta = 0.019$). This pattern is consistent with the intuition that resampling yields the largest gains when baseline uncertainty is higher models that are already confident have less room for improvement via resampling.

4.3 Per-Model Analysis

GPT-5. Using GPT-5 with $\theta = 0.3$ and $N = 3$, self-correction was triggered on 101 of 500 tasks (20.2%). GPT-5 achieved pass@1 $= 0.756$. When resampling was triggered, mean uncertainty decreased from 0.494 to 0.374 (mean drop of 0.120), demonstrating that resampling systematically reduces uncertainty on the most uncertain instances. The overall mean uncertainty decreased from 0.237 to 0.213. This reliability gain comes at additional latency: mean per-task runtime was 37.6s ($p_{95} = 163.4$s), with non-corrected tasks averaging 12.4s and corrected tasks averaging 136.8s (a $\sim 3.0\times$ multiplier). These results validate that uncertainty triggered resampling is used sparingly yet consistently reduces trajectory-level uncertainty.

Claude Opus 4.5. Claude Opus 4.5 completed all 500 tasks, with resampling triggered in only 76 cases (15.2%). Mean uncertainty decreased from 0.230 to 0.211, and average end-to-end latency was 18.3s per task the lowest of all three models. Claude's low resampling rate reflects its more stable trajectory-level confidence. The threshold-based policy adds minimal overhead for Claude, making it the most compute-efficient model under SAUP.

DeepSeek V3.2. DeepSeek V3.2 completed 498 tasks successfully (with 2 error records), with resampling triggered in 139 cases (27.9%). DeepSeek achieved pass@1 $= 0.739$. Mean uncertainty decreased from 0.289 to 0.230 ($\Delta = 0.059$), the largest reduction among all models. Average latency was 84.6s, reflecting both the higher resampling rate and the multi-step agent interaction cost. DeepSeek benefits most from SAUP-driven self-correction, as its higher baseline uncertainty provides more room for improvement.

5 Limitations and Future Work

Limitations. Our uncertainty estimation relies primarily on API-level signals (`finish_reason` metadata), which provide limited discriminative power (AUROC ≈ 0.5); Claude Opus 4.5's near-constant confidence scores illustrate a worst case of this approach. Additionally, the optimal threshold $\theta = 0.3$ was tuned on a single benchmark (SWE-Rebench), and generalization to other agentic settings web navigation, robotics, or safety-critical domains remains unvalidated. The causal mechanism behind the improvement is also not fully disentangled: gains may arise from directly correcting errors, from the selection effect of choosing the lowest-uncertainty candidate, or both.

Future work. Richer uncertainty signal such as semantic entropy from multiple completions, ensemble disagreement, or model-internal representations could substantially improve calibration. On the resampling side, partial trajectory resampling (restarting only from the high-uncertainty step rather than from scratch) and learned budget allocation policies that spend more compute on harder tasks are promising directions for reducing latency overhead. Extending SAUP-driven self-correction to the broader AgentBench suite (Liu et al., 2023) and integrating it with tree-search or Monte Carlo trajectory methods are natural next steps.

6 Author Contributions

JA implemented and evaluated the adapted SAUP uncertainty quantification on SWE-Rebench and contributed to writing. **LSD** and **AP** Conducted the self-correction and resampling experiments and contributed to writing. **SW** conceived the original idea and advised the project. **TA** led the project, designed the experimental framework, provided technical guidance, and contributed to writing and editing the manuscript. **SW** and **RL** provided advisory feedback throughout the project and reviewed the manuscript.

References

Anthropic. Claude opus 4.5 model card. 2025. URL `https://www.anthropic.com`.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents, 2025. URL `https://arxiv.org/abs/2505.20411`.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue

Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Shubham Gandhi, Jason Tsay, Jatin Ganhotra, Kiran Kate, and Yara Rizk. When agents go astray: Course-correcting SWE agents with PRMs. *arXiv preprint arXiv:2509.02360*, 2025. URL https://arxiv.org/abs/2509.02360.

Kaifeng He, Mingwei Liu, Chong Wang, Zike Li, Yanlin Wang, Xin Peng, and Zibin Zheng. Towards better code generation: Adaptive decoding with uncertainty guidance, 2025. URL https://arxiv.org/abs/2506.08980.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. volume 43, pp. 1–55. Association for Computing Machinery (ACM), January 2025. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

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.

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

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. *arXiv preprint arXiv: 2308.03688*, 2023.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. 2021.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan,

Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan Liss, Jos Kraaijeveld, Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll, Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng, Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan,

Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang Wang. Openai gpt-5 system card, 2025. URL https://arxiv.org/abs/2601.03267.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. 2023.

Zhiqiu Xia, Jinxuan Xu, Yuqian Zhang, and Hang Liu. A survey of uncertainty estimation methods on large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21381–21396, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1101. URL https://aclanthology.org/2025.findings-acl.1101/.

Qiwei Zhao, Dong Li, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, Chen Zhao, Haifeng Chen, and Xujiang Zhao. Uncertainty propagation on LLM agent. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6064–6073, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.302. URL https://aclanthology.org/2025.acl-long.302/.

Yuqi Zhu, Ge Li, Xue Jiang, Jia Li, Hong Mei, Zhi Jin, and Yihong Dong. Uncertainty-guided chain-of-thought for code generation with llms, 2025. URL https://arxiv.org/abs/2503.15341.

A Full Grid Search Results

Table 5 reports the full grid search over (θ, N) for GPT-5 on SWE-Rebench. The optimal configuration is $\theta = 0.3$, $N = 8$ (pass@1 $= 0.955$). Lower thresholds ($\theta = 0.3$) consistently outperform higher thresholds across all values of N, confirming that aggressive resampling on uncertain trajectories is beneficial. Performance degrades substantially at $\theta \geq 0.5$, as the threshold becomes too permissive and fails to trigger correction on trajectories that would benefit from it.

Table 5: Full grid search results for GPT-5 on SWE-Rebench. Each cell reports pass@1.

$\theta \setminus N$	$N = 1$	$N = 3$	$N = 5$	$N = 8$
$\theta = 0.3$	0.720	0.735	0.915	**0.955**
$\theta = 0.5$	0.560	0.535	0.590	0.640
$\theta = 0.7$	0.515	0.575	0.620	0.580
$\theta = 0.9$	0.550	0.565	0.535	0.650

Note: Due to computational constraints, the full (θ, N) grid search was completed only for GPT-5. For Claude Opus 4.5 and DeepSeek V3.2, we evaluated the main configuration ($\theta = 0.3$, $N = 3$) based on the finding that $\theta = 0.3$ is consistently optimal across the GPT-5 grid.

B Confidence Distribution Across Models

Figure 3 shows the mean step confidence across SWE-Rebench samples. GPT-5 maintains the highest mean confidence ($\bar{c} \approx 0.85$), DeepSeek V3.2 shows moderate confidence with slight variation ($\bar{c} \approx 0.80$), and Claude Opus 4.5 exhibits near-constant confidence due to its reliance on fixed API signals.

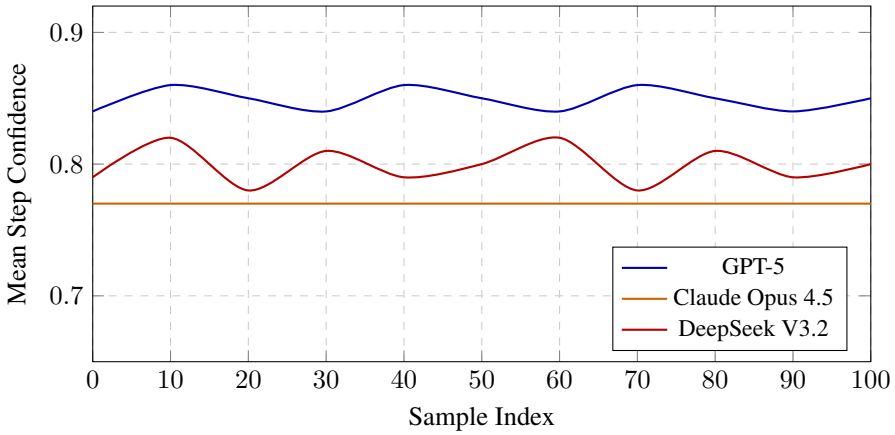

Figure 3: Mean step confidence across SWE-Rebench samples. GPT-5 maintains the highest confidence. Claude Opus 4.5 shows constant confidence due to fixed API signals. DeepSeek V3.2 shows moderate confidence with slight variation.