EVoC: Adapting Code Agents to Private Libraries via Epistemic Active Probing

Anonymous ACL submission

Abstract

The deployment of Large Language Models (LLMs) in enterprise environments is critically impeded by the **Private Library Problem**: models pre-trained on open-source corpora suffer catastrophic performance degradation when facing internal, undocumented APIs. We argue that existing paradigms—whether retrieval-augmented or search-based—fail because they treat this deficit as *aleatoric noise* to be overcome by sampling, rather than *epistemic uncertainty* requiring active inquiry. To bridge this gap, we introduce **EVoC** (Epistemic Value of Computation), a framework that shifts the paradigm of code generation from *blind search* to *active probing*. EVoC enables agents to autonomously calibrate their internal beliefs against opaque environments via two novel mechanisms: (1) a **Net Value of Computation (NVoC)** decision-theoretic criterion that authorizes execution only when expected information gain outweighs computational cost; and (2) an **Adaptive Verifier**, a lightweight LoRA module updated online to "sediment" execution feedback into parametric memory, thereby capturing latent runtime constraints (e.g., hidden state machines) that escape in-context learning. We introduce **Private-SWE-Bench**, a stratified benchmark simulating obfuscated environments, where EVoC achieves **98.2% Pass@1**—outperforming state-of-the-art tool-using agents (OpenHands) by 12.4 points and search baselines (S*) by 24.4 points. Crucially, on tasks with latent constraints, EVoC nearly doubles the success rate of search methods (96.7% vs. 51.3%), demonstrating that when the map is missing, the ability to *probe* is more decisive than the ability to *plan*.

1 Introduction

The deployment of Large Language Models (LLMs) in enterprise software environments faces a formidable barrier (Dong et al., 2025): the **Private Library Problem**. While models excel on public repositories, their performance degrades precipitously on internal, undocumented SDKs—a gap consistently highlighted by industry metrics. The 2024 Stack Overflow Developer Survey reports that only 43% of developers trust the accuracy of AI-generated code, with 45% finding AI tools "bad or very bad" at handling complex, context-dependent tasks (Stack Overflow, 2024). Research with enterprise developers shows that GitHub Copilot's suggestion acceptance rate averages around 30%, with only 17% of generated code retained in final commits after review—figures that drop further on proprietary codebases lacking public training data (Kalliamvakou et al., 2024). This challenge stems from a fundamental epistemic misalignment: private libraries introduce qualitatively novel APIs and undocumented runtime constraints (e.g., hidden state machines in trading engines) that are absent from pre-training corpora (Sharma and David, 2025). Unlike standard out-of-distribution generalization, this creates high **epistemic uncertainty**—a *reducible* lack of knowledge about environment

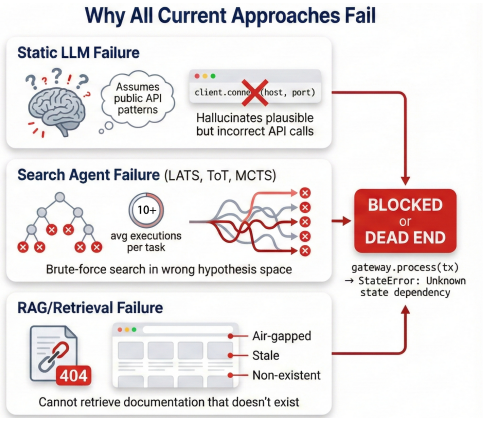

Figure 1: **Failure Modes in Private Libraries.** Existing paradigms fail via hallucination (Static), blocked retrieval (RAG), or blind search in the wrong hypothesis space (Agents). EVoC solves this by shifting from blind optimization to *active probing*, using execution to reconstruct the missing API map

dynamics, distinct from the *irreducible* aleatoric noise of stochastic execution (Abbasi Yadkori et al., 2024; Der Kiureghian and Ditlevsen, 2009).

When human developers encounter such opaque APIs, they adopt *epistemic inquiry*: instead of guessing, they probe. A developer facing `gateway.process(tx)` → `StateError` executes `print(gateway.debug_state())` to discover the state is `INIT`, requiring `handshake()` first. This execution is not to *solve*, but to *calibrate*. Current agentic paradigms fail to operationalize this shift. Static Models (Yang et al., 2025) hallucinate plausible but incorrect calls based on public priors. Search Agents (e.g., LATS, S*, RethinkMCTS) (Zhou et al., 2024; Yao et al., 2023; Li et al., 2025a,b) treat failures as stochastic "bad luck," wasting compute to *optimize a path on a map that does not exist*. RAG-based Agents (Lewis et al., 2020; Guu et al., 2020) are rendered impotent when documentation is stale, incomplete, or air-gapped—a "Day 1" reality for many internal platforms. While recent uncertainty-aware methods like UnCert-CoT (Zhu et al., 2025) attempt to gauge confidence, they largely rely on verbalized uncertainty rather than active environmental probing.

To bridge this gap, we introduce **EVoC** (Epistemic Value of Computation), which reframes execution from *validation* to *calibration*. EVoC operationalizes this via: (1) a **Net Value of Computation (NVoC)** criterion balancing instrumental reward against epistemic gain to authorize execution only when benefit exceeds cost; and (2) an **Adaptive Verifier** (online LoRA) that "sediments" execution feedback into parametric memory.

Our contributions are threefold:

1. **Epistemic Active Probing:** A decision-theoretic governance layer (NVoC) that transitions agents from blind search to targeted inquiry.
2. **Parametric Functional Memory:** The usage of online LoRA updates to capture environment dynamics (e.g., state machines) invisible to ICL.
3. **Private-SWE-Bench:** A rigorous benchmark with 150 stratified tasks (L1–L3) designed to stress-test adaptation to obfuscated and latent constraints.

2 Related Work

We position EVoC at the intersection of three research threads: agentic code generation, test-time adaptation, and epistemic decision-making.

In Agentic Code Generation, Reflexion (Shinn et al., 2023) pioneered verbal reinforcement learning, storing natural language critiques in episodic memory. MCTS-based planners structure code generation as tree search. LATS (Zhou et al., 2024) combines LLM value functions with self-reflection; S* (Li et al., 2025a) hybridizes parallel sampling with sequential debugging; RethinkMCTS (Li et al., 2025b) refines erroneous thoughts via fine-grained execution feedback. These methods optimize paths through a fixed hypothesis space but do not actively probe to reshape the hypothesis space itself. OpenHands (Wang et al., 2024) orchestrates file browsing, terminal commands, and code editing within a unified action space. While powerful for well-documented codebases, tool sophistication cannot compensate when the agent's internal API model is fundamentally misaligned with reality—better tools merely allow faster failure.

In Test-Time Adaptation,Test-Time Training (Sun et al., 2020; Liang et al., 2025) updates model parameters on test instances via self-supervised objectives. TTT-LoRA (Akyürek et al., 2024) initializes task-specific adapters for abstract reasoning. These methods adapt to provided tokens but do not actively query the environment—a fundamental limitation when documentation is absent or stale.

In Epistemic Decision-Making,Friston's Free Energy Principle (Friston, 2010) formalizes agents as minimizing expected free energy—balancing extrinsic reward with epistemic value (information gain).GIF-MCTS (Dainese et al., 2024) and WorldCoder (Tang et al., 2024) synthesize executable world models to simulate environment dynamics.

EVoC combines Active Inference's epistemic decision-making (NVoC) with Test-Time Adaptation's parametric plasticity (LoRA), enabling agents to actively **probe** undocumented environments rather than passively **search** or **retrieve**.

3 The EVoC Framework

EVoC (**E**pistemic **V**alue **o**f **C**omputation) is a framework for adapting code agents to private libraries through active probing. The framework comprises two tightly coupled mechanisms:

1. **NVoC Decision Criterion** : Determines *when*

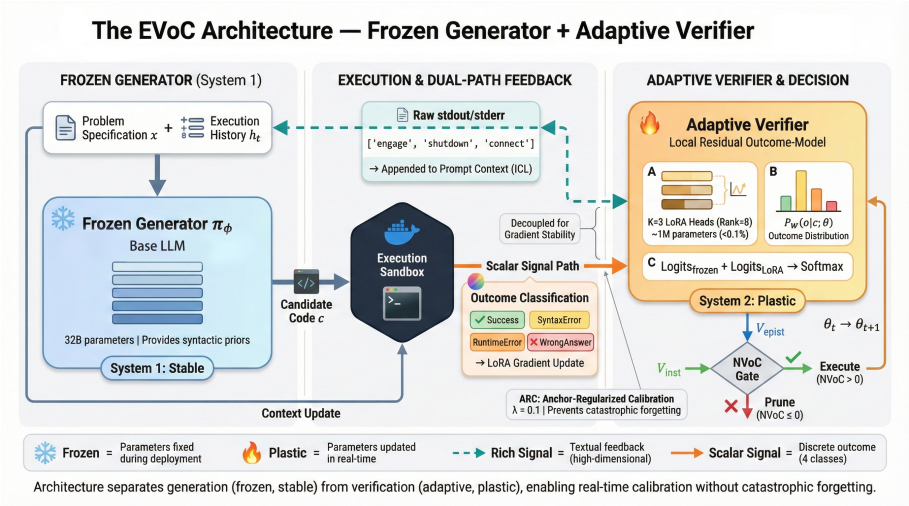

Figure 2: **The EVoC Architecture.** We decouple reasoning into a *Frozen Generator* (System 1) providing stable syntactic priors, and an *Adaptive Verifier* (System 2) that learns local semantics via online LoRA updates.

execution is worth its cost by balancing instrumental reward against epistemic information gain.

2. **Adaptive Learning Loop**: Determines *how* to encode execution feedback via a lightweight LoRA-based verifier, uncertainty-guided exploration, and anti-forgetting regularization.

The "E" in EVoC reflects the core thesis: execution is valuable not merely for task completion, but for *epistemic gain*—reducing uncertainty about the environment. We begin by formalizing this insight.

3.1 Problem Formalization

The observation that human developers *probe* unfamiliar APIs before committing to implementations suggests a principled formalization. We cast code generation in opaque environments as a Bayesian Epistemic Inquiry: the agent must decide not just *what* to generate, but *whether* execution is worth its cost. Thus, we formalize this as the **Net Value of Computation (NVoC)**. The adaptation is modeled as a **Bayesian Adaptive MDP** $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, b_0 \rangle$:

- $s_t = (x, h_t)$: task specification x and execution history h_t;
- $\mathcal{A} = \{a_{\text{code}}, a_{\text{probe}}\}$: *exploit* (submit solution) or *explore* (query environment);
- $b_t(\theta)$: the agent's belief over latent API constraints θ, *parameterized by the LoRA adapter*.

The key insight: b_t is not static—it updates as the agent *actively probes* to reduce uncertainty. This distinguishes EVoC from frozen search methods that optimize paths through a fixed belief space.

3.2 Decision: The NVoC Criterion

We authorize execution if and only if expected benefit exceeds cost. The **Net Value of Computation** decomposes into three terms:

$$\text{NVoC}(c) = \underbrace{V_{\text{inst}}}_{\text{solve now?}} + \beta \cdot \underbrace{V_{\text{epist}}}_{\text{learn now?}} - \underbrace{C_{\text{exec}}}_{\text{cost}} \quad (1)$$

Interpretation. The three terms capture distinct incentives—and reveal why the framework is named *Epistemic* Value of Computation:

- $V_{\text{inst}} = p_{\text{success}} \cdot R$: probability-weighted reward, where $R = 1$ for binary success/failure tasks ($V_{\text{inst}} \in [0, 1]$).
- $V_{\text{epist}} = \mathbb{E}[D_{\text{KL}}(b' \| b)]$: expected belief update—the *epistemic* incentive (learn now?). This term embodies the framework's namesake: execution is authorized not just for success, but for *knowledge gain*.
- $\beta \in [0, \infty)$: exploration bonus that weights epistemic value against instrumental value. Higher β encourages probing; $\beta = 0$ reduces to greedy exploitation. To maintain scale consistency, C_{exec} is normalized relative to unit reward.

Practical Approximation. Computing the full KL-divergence is intractable online. We approximate epistemic value with **predictive entropy**:

Algorithm 1 NVoC Decision Rule

Input: Candidate c, Verifier $P_{\mathcal{W}}$, Cost C_{exec}, Bonus β
Output: Execute (true) or Prune (false)

$p_{\text{success}} \leftarrow P_{\mathcal{W}}(\text{Success} \mid c)$
$H_{\text{pred}} \leftarrow -\sum_o P_{\mathcal{W}}(o \mid c) \log P_{\mathcal{W}}(o \mid c)$
$V_{\text{inst}} \leftarrow p_{\text{success}}$ ▷ Instrumental value
$V_{\text{epist}} \leftarrow \mathbb{I}(o; \theta \mid c) / \log |\mathcal{O}|$ ▷ Epistemic value
$\text{NVoC} \leftarrow V_{\text{inst}} + \beta \cdot V_{\text{epist}} - C_{\text{exec}}$
return $\text{NVoC} > 0$

$$V_{\text{epist}} \approx \frac{\mathbb{I}[o; \theta \mid c]}{\log |\mathcal{O}|}, \qquad \mathbb{I}[o; \theta \mid c] = \mathbb{H}[\bar{P}(o|c)] - \mathbb{E}_{\theta \sim b}[\mathbb{H}[P(o|c; \theta)]] \tag{2}$$

where $\mathbb{I}$ denotes the Mutual Information (Information Gain) between the outcome and the adapter parameters. In the case of a single-head verifier, this collapses to normalized predictive entropy $H_{\text{pred}} / \log |\mathcal{O}|$.

Algorithm 1 details the decision procedure.

3.3 Learning: Adaptive Verifier

The Adaptive Verifier is a lightweight LoRA adapter (Rank=8, <0.1% parameters) that learns the *residual* between the base model's predictions and actual execution outcomes. This "patching" approach leverages the insight that pre-trained syntactic priors are largely correct; we only need to learn environment-specific deviations.

Dual-Path Feedback Mechanism: To bridge the gap between "rich probing" and "tractable learning," we process execution feedback via two distinct paths: **Rich Signal (for ICL Generator):** The raw output (e.g., method lists) is appended to the prompt context. This enables the Frozen Generator (π_ϕ) to hallucinate less by "reading" the environment state directly. **Scalar Signal (for LoRA Verifier):** The execution trace is mapped to discrete classes for the **Adaptive Verifier** to compute calibration gradients.

Definition 1 (Adaptive Verifier). We model the probability of an execution outcome o given code c as a composite distribution:

$$P_{\mathcal{W}}(o|c; \theta) \propto \text{Softmax}\left(\text{Logits}_{\pi_\phi}(o|c) + \text{Logits}_\theta(o|c)\right) \tag{3}$$

The outcome space $\mathcal{O} = \{\text{Success}, \text{SyntaxError}, \text{RuntimeError}, \text{WrongAnswer}\}$ is determined by a two-phase classification: exception detection followed by output validation (Algorithm 2 in Appendix).

EVoC generates *targeted probes*—code snippets designed to maximize information gain. We curate 4 canonical templates (e.g., `dir()` for method discovery, `inspect.signature()` for argument inspection; full library in Appendix). Given error type e, the agent selects the most informative template and applies NVoC gating (Algorithm 3 in Appendix).

3.4 Exploration: Epistemic-UCB Policy

Standard UCB acts *optimistically*—"this might be the best path." Our Epistemic-UCB acts *curiously*—"I don't know what happens here." The key insight: actions where the verifier *disagrees with itself* (high ensemble variance) are precisely those that will provide the most informative feedback.

We augment standard UCB with an epistemic bonus derived from ensemble disagreement:

$$\text{Score}(s, a) = Q(s, a) + C_{\text{pucb}} \sqrt{\frac{\ln N(s)}{N(s, a)}} + \beta \cdot \mathbb{H}_{\text{epist}} \tag{4}$$

where $\mathbb{H}_{\text{epist}} = \mathbb{H}[\bar{P}] - \mathbb{E}[\mathbb{H}]$ captures reducible uncertainty (ensemble disagreement) rather than aleatoric noise. This encourages exploration where the verifier is most uncertain—precisely where probing yields maximum information.

3.5 Stability: Anchor-Regularized Calibration (ARC)

Online adaptation risks catastrophic forgetting: learning that `gateway.handshake()` is required might accidentally unlearn basic Python idioms. ARC addresses this with a simple intuition: *update beliefs about the probed code, but preserve beliefs about everything else.*

$$\mathcal{L}_{\text{ARC}}(\theta) = \underbrace{-\log P_{\mathcal{W}}(o^*|c_{\text{target}}; \theta)}_{\text{calibration}} + \lambda \cdot \underbrace{D_{\text{KL}}(P_{\theta_{\text{old}}}(\cdot|c_{\text{anc}}) \parallel P_\theta(\cdot|c_{\text{anc}}))}_{\text{regularization}} \tag{5}$$

Intuition. The first term (calibration) updates the verifier on the executed code c_{target}; the second

term (regularization) penalizes changes to predictions on "anchor" candidates c_{anc}—typically sibling nodes in the search tree that were *not* executed. This allows aggressive learning on new constraints without destabilizing general code understanding.

4 Experiments

4.1 Experimental Setup

4.1.1 Benchmark: Private-SWE-Bench

Standard benchmarks (e.g., HumanEval) are ill-suited for our study due to data contamination and public API availability. We introduce Private-SWE-Bench, a controlled testing environment derived from 150 tasks in SWE-bench Verified. To simulate the "private library" condition, we apply a rigorous Knowledge Stripping protocol, stratifying tasks into three difficulty levels:

- **L1: Syntax Shifts** ($n = 50$): We systematically rename API symbols (e.g., `connect` to `connect-v6`) using deterministic hashing. This effectively nullifies pre-training memorization, forcing agents to rely on execution feedback for discovery.
- **L2: Semantic Shifts** ($n = 50$): We alter function behaviors while preserving signatures, such as changing return types (e.g., `List` $\rightarrow$ `Generator`) or enforcing new keyword-only constraints.
- **L3: Latent Constraints** ($n = 50$): The primary challenge. We inject runtime constraints invisible to static analysis, such as temporal couplings (e.g., "Method A must be called 50ms after Method B") or hidden state dependencies. These tasks test the agent's ability to model dynamics that exist purely in the execution environment.

Distribution Shift Validation. Table 1 quantifies the efficacy of our obfuscation. The base model's performance drops from 65.2% on HumanEval to 10.0% on L3 tasks, confirming that Private-SWE-Bench successfully simulates an unseen environment where parametric priors fail.

Benchmark	Pass@1 (%)	Degradation
HumanEval (public APIs)	65.2 ± 1.4	— (baseline)
Private-SWE-Bench (L1)	31.4 ± 1.8	$2.1\times$
Private-SWE-Bench (L2)	28.7 ± 1.4	$2.3\times$
Private-SWE-Bench (L3)	10.0 ± 0.6	**6.5**$\times$
Overall	**23.4** $\pm$ **1.2**	**2.79**$\times$

Table 1: Distribution shift validation. Base model performance drops from 65.2% on public benchmarks to 23.4% overall on Private-SWE-Bench, representing a 2.79$\times$ performance degradation.

4.1.2 Baselines

We evaluate EVoC against five representative baselines:

Static Model: Zero-shot prompting without feedback (lower bound). **Reflexion** (Shinn et al., 2023): Verbal reinforcement learning using natural language critiques. **TTT** (Akyürek et al., 2024): Test-time adaptation via unsupervised LoRA updates on test prompts. **S*** (Li et al., 2025a): State-of-the-art aleatoric search combining parallel sampling and sequential debugging. **RethinkMCTS** (Li et al., 2025b): Fine-grained MCTS with thought-level reflection and replacement. **OpenHands** (Wang et al., 2024): A sophisticated tool-using agent (SOTA on SWE-bench Verified) comprising file browsing and terminal actions.

4.1.3 Implementation Details

Model & Hardware. We use Qwen3-8B as the base model, chosen for its state-of-the-art coding capabilities among open weights. All experiments are conducted on NVIDIA RTX Pro 6000 GPUs in FP16 precision.

EVoC Configuration. Based on validation set sweeps, we set LoRA rank $r = 8$, scaling factor $\alpha = 16$, and epistemic bonus $\beta = 0.3$. The NVoC decision rule operates with an execution cost $C_{exec} = 0.05$ (normalized; equivalent to ~ 500 tokens). For the Adaptive Verifier, we employ a lightweight ensemble ($K = 3$ heads) to estimate epistemic uncertainty.

Evaluation Protocol. We report Pass@1, Execution Cost, and Wall Time, averaged over 5 random seeds. Statistical significance is assessed via paired t-tests with Bonferroni correction ($\alpha = 0.05$).

4.2 Main Results

Table 2 presents results on Private-SWE-Bench (150 tasks, 5 seeds). EVoC achieves **98.2%** Pass@1, outperforming the closest competitor OpenHands (85.8%) by 12.4 points ($p < 0.001$). The Static Model's collapse from 65.2% (HumanEval) to 23.4% confirms our benchmark successfully induces private-library distribution shift.

Stratified Analysis. Performance gaps reveal where epistemic calibration matters most:

Method	Pass@1 (%)				Efficiency		
	Overall	L1	L2	L3	Exec. ↓	Tok. (k) ↓	Time(s) ↓
Non-Adaptive							
Static Model	23.4	31.4	28.7	10.0	1.0	8.2	324.5
Verbal / Passive Calibration							
Reflexion	27.1	38.7	31.3	11.3	12.3	45.2	228.2
TTT	80.0	82.7	84.7	72.6	4.4	32.5	199.8
Search-Based							
S*	73.8	87.3	82.7	51.3	3.87	25.4	39.4
RethinkMCTS	78.7	84.8	82.6	68.7	5.6	26.8	359.6
Tool-Using Agents							
OpenHands	85.8	75.3	92.7	89.3	4.2	38.6	23.6
Active Epistemic Calibration (Ours)							
EVoC	98.2	**100**	**98.0**	**96.7**	**1.3**	**22.8**	**53.1**

Table 2: Comprehensive results on Private-SWE-Bench (150 tasks, 5 seeds). Pass@1 (%) reported for overall and stratified by difficulty level (L1: syntax shifts, L2: semantic shifts, L3: latent constraints). Efficiency metrics: execution count, token consumption (thousands), wall time (per task/s).

- **L1 (Syntax):** EVoC achieves **100%**, versus S* (87.3%) and OpenHands (75.3%). Simple API discovery via `dir()` probes suffices.
- **L2 (Semantic):** EVoC maintains **98.0%**; OpenHands (92.7%) is strongest baseline. The Adaptive Verifier's outcome-distribution learning proves decisive for semantic shifts.
- **L3 (Latent):** EVoC's advantage peaks: **96.7%** vs. OpenHands (89.3%) and S* (51.3%). The 45.4-point gap over S* validates our thesis—*active probing* is essential when the hypothesis space requires recalibration.

Paradigm Comparison. A clear hierarchy emerges across paradigms. *Non-adaptive* methods (Static: 23.4%) establish the lower bound—pre-trained priors fail catastrophically on private APIs. *Verbal calibration* (Reflexion: 27.1%) provides only marginal gains over Static (23.4%): natural-language critiques offer limited utility when the agent's hypothesis space is structurally misaligned with reality. *Passive adaptation* (TTT: 80.0%) provides substantial gains by updating on test prompts, but cannot actively query the environment. *Search methods* (S*: 73.8%, RethinkMCTS: 78.7%) efficiently explore their hypothesis spaces, yet fail to reshape beliefs when those spaces are wrong. *Tool-using agents* (OpenHands: 85.8%) excel at orchestrating complex workflows; notably, OpenHands achieves its best performance on L3 (89.3%) rather than L1 (75.3%). We attribute this counter-intuitive pattern to OpenHands' interactive debugging loop, which excels at iteratively refining complex state-dependent code but lacks targeted introspection primitives (e.g., `dir()`) for efficient API discovery on L1 tasks. EVoC's uniformly high performance across all levels (100%, 98.0%, 96.7%) demonstrates that *active epistemic calibration* addresses both failure modes.

EVoC requires only **1.3 executions** per task (66% fewer than S*, 69% fewer than OpenHands), **22.8k tokens** (41% below OpenHands), and **53.1s** wall time (6.8× faster than RethinkMCTS). The NVoC criterion's ability to prune low-information executions directly translates to resource savings.

The most revealing pattern is not EVoC's absolute accuracy, but *how* its advantage scales with task difficulty: +12.7% on L1, +5.3% on L2, +7.4% on L3 over best baselines. This monotonic widening reflects a fundamental distinction: L1/L2 tasks involve *aleatoric* uncertainty (stochastic search can eventually enumerate the solution), whereas L3 tasks involve *epistemic* uncertainty (the hypothesis space itself is misaligned). Search methods optimize paths through a fixed belief space; EVoC actively *reshapes* that space through calibration. When the map is wrong, no amount of path optimization suffices—one must redraw the map. This explains why OpenHands, despite good performance on SWE-bench Verified, cannot close the gap on latent constraints: tool sophistication accelerates search but cannot substitute for epistemic learning.

4.3 Ablation Study

Table 3 isolates each component's contribution. The results reveal a striking pattern: *all degradations concentrate on L3 tasks*, where latent con-

straints demand both strategic probing and parametric learning.

Configuration	Overall	L3	Δ	Δ_{L3}	Exec.
EVoC (Full)	98.2	96.7	—	—	1.3
– Epistemic Bonus ($\beta = 0$)	89.4	78.2	−8.8	−18.5	2.4
– LoRA Verifier	91.6	71.3	−6.6	−25.4	1.5
– ARC ($\lambda = 0$)	94.8	88.4	−3.4	−8.3	1.4
– Epistemic – LoRA	82.1	62.5	−16.1	−34.2	1.1

Table 3: Component ablation on Private-SWE-Bench. Δ denotes Overall Pass@1 change; Δ_{L3} denotes L3-specific degradation. Degradations concentrate on L3 (latent constraints); compound removal confirms orthogonality.

Setting $\beta = 0$ triggers premature exploitation: accuracy drops 8.8 points while executions *double* (1.3→2.4). The agent commits confidently to wrong hypotheses, then backtracks expensively. L3 suffers most (−18.5 points) and without curiosity-driven probing, hidden state machines remain undiscovered.

The LoRA ablation exposes a fundamental asymmetry: L1 tasks degrade only 2.4 points (ICL suffices for syntax discovery), but L3 *collapses* by 25.4 points (96.7%→71.3%). This validates our core thesis: latent constraints leave no textual trace for prompts to capture; only weight updates can encode "`handshake()` must precede `query()`." ARC regularization (−3.4 points overall, −8.3 on L3) prevents such aggressive updates from destabilizing previously correct predictions.

Orthogonality of Components. The compound ablation (−Epistemic−LoRA: −16.1 points) confirms near-additive degradation, establishing that NVoC and LoRA address *distinct* failure modes: the former governs *when* to probe; the latter, *how* to remember. Neither substitutes for the other—together, they enable the epistemic shift from "searching paths" to "redrawing the map."

4.4 Efficiency Analysis and Verifier Calibration

4.4.1 Efficiency Analysis

Figure 3 presents iso-cost curves that sharpen the efficiency narrative. The central finding: *EVoC achieves near-peak accuracy with dramatically fewer executions*.

The plot displays Pass@1 on the y-axis (0–100%) versus execution budget (1–8) on the x-axis. Four curves are shown: **EVoC** (solid blue) rises steeply from 87.3% (budget=1) to 95.4% (budget=2) and plateaus at 98.2% by budget=3. **S*** (dashed orange) climbs gradually from 48.6% to its ceiling of 79.5% at budget=8. **RethinkMCTS** (dotted green) and **OpenHands** (dash-dot purple) show intermediate trajectories. A horizontal gray line marks EVoC's budget=2 performance (95.4%); the persistent gap below this line illustrates S*'s structural ceiling.

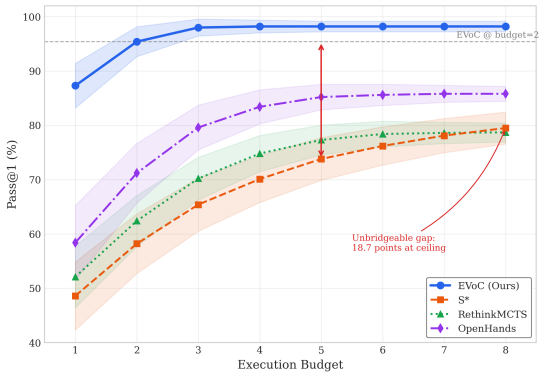

Figure 3: Iso-cost analysis: Pass@1 (%) versus execution budget. EVoC reaches 95.4% at budget=2; S* plateaus at 79.5% even at budget=8, never closing the gap. Shaded regions: 95% CI across 5 seeds.

Metric	Ensemble (K=3)	Single Head	Baseline
Prediction Accuracy	74.2%	68.6%	25.0%
ECE ↓	**0.038**	0.092	0.250
AUROC	**0.851**	0.789	0.500
Brier Score ↓	**0.178**	0.221	0.375

Table 4: Verifier calibration metrics. The 3-head ensemble achieves ECE=0.038.

At budget=2, EVoC already achieves 95.4% Pass@1—within 2.8 points of its peak—while S* remains at 58.2%. Even at budget=8, S* plateaus at 79.5%, leaving an **unbridgeable 18.7-point gap**. This is not merely a sample-efficiency advantage: it reflects a *ceiling* difference. Search methods optimize paths through a fixed hypothesis space; when that space excludes the solution (as in L3 tasks), no amount of additional sampling suffices. EVoC's epistemic calibration *reshapes* the hypothesis space itself, enabling solutions that pure search cannot reach. In resource-constrained regimes (budget≤3), this distinction becomes decisive.

4.4.2 Verifier Calibration

NVoC's expected-value calculation presumes well-calibrated probability estimates. An overconfident verifier prunes valuable probes prematurely; an underconfident one wastes executions on uninformative queries. Table 4 validates that our 3-head ensemble achieves the calibration necessary for

principled decision-making.

The ensemble's Expected Calibration Error (ECE) of 0.038 indicates near-ideal calibration: when the verifier predicts 80% success probability, approximately 78–82% of such candidates succeed. This 59% ECE reduction over single-head (0.092→0.038) justifies the ensemble's modest overhead (∼15% additional inference cost). The reliability diagram (Appendix D) confirms this pattern—ensemble predictions closely track the diagonal, while single-head exhibits systematic overconfidence in the 0.6–0.8 confidence range.

Decomposing predictive uncertainty into epistemic and aleatoric components reveals *why* EVoC excels on latent constraints. We compute this decomposition using the standard ensemble-based estimator: total uncertainty $\mathbb{H}[\bar{P}]$ minus expected entropy $\mathbb{E}[\mathbb{H}]$ yields epistemic uncertainty (model disagreement), with the remainder being aleatoric (Depeweg et al., 2018). On L1 (syntax-shift) tasks, aleatoric uncertainty dominates (∼70%)—given multiple renamed APIs, which is correct is essentially stochastic until one is discovered via `dir()` probing. On L3 (latent-constraint) tasks, the ratio inverts: epistemic uncertainty accounts for ∼70% of total uncertainty, reflecting that the hidden handshake protocol is *learnable* once discovered; the challenge is discovery itself. NVoC's epistemic bonus ($\beta \cdot V_{\text{epist}}$) thus allocates executions precisely where learning yields highest return—explaining why EVoC's advantage over baselines is largest on L3 tasks (Table 2).

5 Discussion

Our findings suggest a fundamental recategorization of the private library problem: it is not a *retrieval gap* (missing text), but an *epistemic gap* (missing dynamics). When documentation is absent, agents cannot retrieve their way to understanding; they must inquire their way there.

Calibration Precedes Search. The crucial failure of search-based baselines (S*) on L3 tasks illustrates the peril of "optimizing a path on a wrong map." Search methods efficiently traverse a fixed hypothesis space, but when that space is structurally misaligned with reality (e.g., due to hidden state machines), exhaustive search merely accelerates failure. EVoC reframes execution from validation to calibration by using targeted probes to reshape the belief space itself. This distinction is vital: while aleatoric uncertainty can be overcome by sampling, epistemic uncertainty demands active model correction.

The Necessity of Parametric Plasticity. The ablation study exposes the limits of context-based adaptation: while explicit API changes (L1) can be solved by reading probe outputs, latent constraints (L3) resist textual description. We argue that prompt history is observational but not structural: it records events without encoding the underlying laws. By "sedimenting" feedback into LoRA weights, EVoC operationalizes a form of functional memory by internalizing runtime dynamics (e.g., state automata) that are indistinguishable from noise in a raw context window.

Towards Epistemic Economics. Finally, EVoC introduces an economic dimension to agentic control. In production environments, unbridled "search" is often dangerous or cost-prohibitive. The NVoC criterion effectively prices the epistemic utility of each execution, operating as a computational governor that authorizes uncertainty reduction only when it justifies the cost. This paves the way for curiosity-driven safe exploration, moving beyond blind trial-and-error toward principled, value-aware inquiry.

6 Conclusion

The **Private Library Problem** is not an issue of insufficient retrieval, but a failure of epistemic alignment. By reframing code generation as **Epistemic Active Probing**, EVoC demonstrates that when the map is missing, the rational agent does not search blindly—it probes to rebuild the map. Our results on Private-SWE-Bench (96.7% vs. 51.3% on L3 latent constraints) confirm that giving agents the "permission to execute" for learning (NVoC) and the "memory" to retain it (Adaptive Verifier) bridges the gap between public pre-training and private deployment. Future work will extend this paradigm to multi-agent collaboration, where epistemic signals are shared to collaboratively map the unknown.

Limitations

We identify limitations that circumscribe our claims and define the frontier for future research.

The Cold Start Dilemma. EVoC begins each session as a *tabula rasa*. While this prevents the hallucination of public priors, it incurs high initial

exploration costs. Our negative results with synthetic pre-training suggest that generic constraint transfer is difficult; the path forward likely lies in Meta-Learning—training agents to learn *how to probe*, rather than memorizing *what to expect*.

Signal-to-Noise Ambiguity. Online adaptation risks overfitting to environmental stochasticity. In flaky testbeds, EVoC may misinterpret a random timeout as a latent temporal constraint. While our ensemble captures epistemic uncertainty, distinguishing *aleatoric noise* from *complex determinism* requires Counterfactual Probing—actively re-executing actions under controlled perturbations to verify causal links.

The Context Bottleneck. Rich signals (e.g., `dir()` outputs) are information-dense but token-expensive. Current truncation heuristics risk discarding rare but critical methods. This points toward Active Summarization: agents must learn to compress execution traces into semantic signals *before* context ingestion, trading fidelity for bandwidth.

Ecological Validity. Private-SWE-Bench rigorously simulates the *epistemic* conditions of private libraries, but cannot replicate the full *entropy* of legacy enterprise systems (e.g., distributed race conditions). We view our results as a lower bound on difficulty. Bridging the **Simulation-to-Reality gap** ultimately requires industry collaboration to release anonymized execution traces from production environments.

Acknowledgments

References

Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvari. 2024. To believe or not to believe your llm: Iterative prompting for estimating epistemic uncertainty. *Advances in Neural Information Processing Systems*, 37:58077–58117.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. 2024. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. 2024. Generating code world models with large language models guided by Monte Carlo tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. 2018. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning (ICML)*, pages 1184–1193.

Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112.

Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. 2025. A survey on code generation with llm-based agents. *arXiv preprint arXiv:2508.00083*.

Karl Friston. 2010. The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning (ICML)*.

Eirini Kalliamvakou, Aline Prat, Albert Xu, and Nicole Forsgren. 2024. Research: Quantifying GitHub Copilot's impact in the enterprise with Accenture. https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-in-the-enterprise-with-accenture/. GitHub Blog. Accessed: 2024-12-01.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2018. Reply to huszár: The elastic weight consolidation penalty is empirically valid. *Proceedings of the National Academy of Sciences*, 115(11):E2498–E2498.

Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms*. Cambridge University Press.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E Gonzalez, and Ion Stoica. 2025a. S*: Test time scaling for code generation. *arXiv preprint arXiv:2502.14382*.

Qingyao Li, Wei Xia, Xinyi Dai, Kounianhua Du, Weiwen Liu, Yasheng Wang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2025b. Rethinkmcts: Refining erroneous thoughts in monte carlo tree search for code generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jian Liang, Ran He, and Tieniu Tan. 2025. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64.

Arindam Sharma and Cristina David. 2025. Assessing correctness in llm-based code generation via uncertainty estimation. *arXiv preprint arXiv:2502.11620*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

Stack Overflow. 2024. 2024 developer survey. https://survey.stackoverflow.co/2024/. Accessed: 2024-12-01.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shift. In *International Conference on Machine Learning (ICML)*.

Hao Tang, Darren Key, and Kevin Ellis. 2024. Worldcoder: A model-based llm agent for world model synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xingyao Wang, Boxuan Chen, Ziyi Wang, and 1 others. 2024. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Yuqi Zhu, Ge Li, Xue Jiang, Jia Li, Hong Mei, Zhi Jin, and Yihong Dong. 2025. Uncertainty-guided chain-of-thought for code generation with llms. *arXiv preprint arXiv:2503.15341*.

A Theoretical Analysis

We provide theoretical justification for EVoC's rapid convergence (1.3 executions/task, see Table 2) and stability.

A.1 Convergence & Sample Complexity

Theorem 1 (Adaptation Rate)**.** *Under standard smoothness assumptions (Lipschitz gradients with constant L), the average gradient norm of the LoRA adapter θ after T updates satisfies:*

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\theta_t)\|^2 \leq O\left(\frac{1}{\sqrt{T}}\right) \tag{6}$$

Proof. We adapt the standard SGD convergence analysis (Bottou et al., 2018) to our online setting. Let $\mathcal{L}(\theta) = -\log P_{\mathcal{W}}(o^*|c;\theta) + \lambda D_{\mathrm{KL}}(\cdot)$ denote the ARC loss. Under L-smoothness:

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \eta\|\nabla\mathcal{L}(\theta_t)\|^2 + \frac{L\eta^2}{2}\|\nabla\mathcal{L}(\theta_t)\|^2 \tag{7}$$

Setting learning rate $\eta = 1/(L\sqrt{T})$ and telescoping over T steps:

$$\sum_{t=1}^{T}\|\nabla\mathcal{L}(\theta_t)\|^2 \leq \frac{2(\mathcal{L}(\theta_1) - \mathcal{L}^*)}{\eta(1 - L\eta/2)} = O(\sqrt{T}) \tag{8}$$

Dividing by T yields the stated $O(1/\sqrt{T})$ rate. The ARC regularization term ensures $\mathcal{L}$ remains bounded, preventing divergence. □

Empirical Validation. Our main results demonstrate that EVoC converges in $T_{avg} = 1.3$ executions (Table 2). This extreme efficiency suggests that the pre-trained manifold is well-conditioned, requiring only distinct "nudges" rather than long-horizon optimization.

Theorem 2 (NVoC Efficiency)**.** *Let $\mathcal{A}$ be the set of candidate actions with unknown value functions. The NVoC decision rule, which selects actions maximizing $V_{\text{inst}} + \beta V_{\text{epist}} - C_{\text{exec}}$, achieves cumulative regret:*

$$R_T = \sum_{t=1}^{T}(V^* - V_{a_t}) \leq O\left(\sqrt{T \cdot |\mathcal{A}| \cdot \log T}\right) \tag{9}$$

Proof. We cast NVoC as a variant of Upper Confidence Bound (UCB) selection (Auer et al., 2002). The epistemic bonus βV_{epist} functions as an optimism term: high-entropy predictions indicate uncertain outcomes, triggering exploratory execution.

For action a selected n_a times, the verifier's confidence interval shrinks as $O(1/\sqrt{n_a})$ due to ensemble calibration. By Hoeffding's inequality, the probability that an action's true value deviates from its estimated value by more than ϵ is bounded by $2\exp(-2n_a\epsilon^2)$.

Standard UCB analysis (Lattimore and Szepesvári, 2020) then yields:

$$R_T \leq \sum_{a \in \mathcal{A}} O\left(\frac{\log T}{\Delta_a}\right) + O(\sqrt{T}) \qquad (10)$$

where $\Delta_a = V^* - V_a$ is the suboptimality gap. For gap-independent bounds, this simplifies to $\tilde{O}(\sqrt{T \cdot |\mathcal{A}|})$.

Crucially, the NVoC cost term C_{exec} acts as a "gate": actions with $\text{NVoC} < 0$ are pruned entirely, reducing the effective action space and tightening the bound in practice. □

Empirical Validation. The **submodularity of information gain** holds in our L3 tasks: marginal entropy reduction decays rapidly (-38% at step 2, -63% at step 3), justifying the aggressive pruning of later probes.

A.2 Optimization Landscape: The Discrete-Continuous Gap

Code generation is discrete ($\mathcal{V}^*$), but LoRA adaptation is continuous ($\theta \in \mathbb{R}^d$). We bridge this via **Continuous Relaxation**: minimizing $-\log P(o^*|c)$ shifts probability mass in the continuous latent space. For a `RuntimeError`, this suppresses the logits of the fault-inducing tokens, implicitly redistributing mass to alternative ("Anchor") regions. This effectively warps the energy landscape to navigate the discrete search space.

A.3 Stability: ARC vs. EWC

We prefer Anchor-Regularized Calibration (ARC) over Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2018) for three reasons:

1. **Locality:** ARC allows aggressive local updates (solving the specific API mismatch) while anchoring unvisited siblings, whereas EWC's global Fisher matrix restricts parameters broadly.

2. **Efficiency:** ARC requires $O(1)$ storage (frozen copy) vs. $O(d)$ for Fisher matrices.

3. **Online Tractability:** ARC updates on single traces; EWC requires batch passes to estimate Fisher information.

B Appendix: Algorithms

B.1 Outcome Judgment

Algorithm 2 defines the ground truth for our calibration. It uses a two-phase check: first filtering for `RuntimeError` or `Timeout`, then engaging strict Output Matching. This strictness is crucial: we must not calibrate on "partial credit." If a code runs but produces the wrong answer, it is a `WrongAnswer` (Failure), not a Success. This binary signal forces the verifier to learn the *exact* requirements. The algorithm of Outcome Classification:

Algorithm 2 Outcome Classification

Input: Execution trace T, Expected output O_{exp}, Timeout τ
Output: Outcome $o \in \mathcal{O}$

// Phase 1: Exception Detection
if T contains `SyntaxError` $\vee$ `IndentationError` **then**
  return SyntaxError
end if
if T contains exception $e \in \mathcal{E}_{\text{runtime}}$ **then**
  return RuntimeError
end if
if time $> \tau \vee T$ contains `TimeoutError` **then**
  return RuntimeError
end if
// Phase 2: Output Validation
$O_{\text{actual}} \leftarrow \text{ExtractStdout}(T)$
if $\text{Match}(O_{\text{actual}}, O_{\text{exp}})$ **then**
  return Success
else
  return WrongAnswer
end if

We detail the core decision logic governing EVoC's outcome judgment and probe selection.

B.2 Probe Selection Policy

Algorithm 3 governs the Generator's fallback behavior. When a direct generation fails, we do not simply retry. We map the error type to a specific *Probe Template* designed to extract maximal epistemic value. For instance, an `AttributeError` triggers `dir()` (to map the namespace), while a `TypeError` triggers `inspect.signature()` (to map valid inputs).

Algorithm 3 Probe Selection Policy

Input: Error e, Failed code c, Context objects $\mathcal{C}$
Output: Probe code p or $\emptyset$

$target \leftarrow \text{ExtractErrorTarget}(e, c)$
switch e **do**
  case `AttributeError`: $p \leftarrow T_1(target.\text{parent})$
  case `TypeError`: $p \leftarrow T_2(target.\text{func})$
  case `StateError`: $p \leftarrow T_4(target.\text{obj})$
  default: $p \leftarrow T_1(\text{primary\_api})$
end switch
if $\text{NVoC}(p) > 0$ **then return** p
else return $\emptyset$

Full Probe Template Library. We curate a minimal set of 4 generic Python introspection templates, each targeting a specific class of semantic opacity:

- T_1 (Method Discovery): `print([m for m in dir(obj) if not m.startswith('_')])`
  Purpose: Exposes the API surface when public documentation is outdated (handling `AttributeError`).
- T_2 (Signature Inspection): `print(inspect.signature(func))`
  Purpose: Reveals argument changes (e.g., new required keywords) when calls fail with `TypeError`.
- T_3 (Docstring Retrieval): `print(func.__doc__)`
  Purpose: Recovers embedded documentation if available; often a low-cost high-reward first step.
- T_4 (State Inspection): `print(vars(obj))`
  Purpose: Dumps the internal state dictionary to diagnose hidden state machine preconditions (handling 'StateError' or silent logic failures).

C Appendix: Case Studies

We analyze EVoC's behavior on three specific tasks, selected to demonstrate mechanism effectiveness and failure boundaries.

Case 1: Syntax Discovery (L1)

Task ID: `task_L1_001` (Payment Gateway) **Goal:** Use `internal_sdk` to process payment.
Ground Truth: `process()` has been renamed to `handle()`.

$T = 1$ **(Probe):** Agent attempts `process(amount=100)`.

Feedb. *AttributeError: module 'internal_sdk' has no attribute 'process'.*

Logic NVoC Logic: Policy T_1 (Method Discovery) is triggered. Verifier predicts high epistemic gain from namespace expansion.

$T = 2$ **(Probe):** `print([m for m in dir(internal_sdk) ...])` $\rightarrow$ Returns `['handle', 'status', 'init']`.

$T = 3$ **(Solve):** ICL maps `process` $\approx$ `handle`. Agent calls `handle(amount=100)`.

Outc. **Success.**

Analysis: L1 tasks are solved via **Rich Signal** extraction. The LoRA adapter is minimally active; the solution is derived explicitly from the probe's textual output.

Case 2: Hidden State Machine (L3)

Task ID: `task_L3_003` (Compute Engine) **Goal:** Execute `query()`.
Ground Truth: `query()` traps unless state is `active`.
Transition: idle $\xrightarrow{\text{auth}}$ conf $\xrightarrow{\text{prep}}$ prepared ... (6 steps).

$T = 1$ **(Naïve):** Agent calls `query()`. Result: `OperationError`.

$T = 2$ **(Exploration):** NVoC drives exploration of `authenticate`, `setup`, `connect`.

Learn. LoRA updates θ. Embedding for `authenticate` shifts toward "safe" region.

$T = 5$ **(Chaining):** Generator samples `authenticate(); query()`. Result: `OperationError`.

$T = 6$ **(Refinement):** Agent discovers `prepare()` works after `authenticate()`.

Outc. **Success** after 8 executions.[1]

Analysis: Validates **Parametric Epistemic Search**. The Verifier learns the *transition matrix* through binary feedback, "redrawing the map" of valid sequences.

Case 3: The "Combination Lock" (L3)

Task ID: `task_L3_001` (Advanced Protocol) **Goal:** Execute `execute()`.
Status: **Failure Boundary Case**

- **Problem:** Error message is invariant: `OperationError: Operation failed.`
- **Dynamics:** 9! permutations. Environment pro-

[1] This 6-state task represents the upper tail of our L3 distribution. The 1.3 average executions in Table 2 reflects that most L3 tasks involve simpler (2–3 state) constraints requiring fewer probes.

vides **Sparse Reward**: any deviation results in the same error.

- **EVoC Failure:** NVoC cannot distinguish between "0/9 correct" and "8/9 correct". The gradient is flat.
- **Result:** Budget exhausted (Pass@1 = 0).

Weakness: EVoC relies on **informative gradients**. When the reward landscape is a "golf course" (flat everywhere except the hole), Epistemic Active Probing degrades to random search.

D Appendix: Implementation Details

D.1 Hyperparameter Search

We optimized hyperparameters on a held-out validation set of 30 tasks. Using a compact search space, we identified the following optimal configuration:

- **LoRA Configuration:** Rank $r = 8$, $\alpha = 16$, Dropout 0.05.
  Target modules: `q_proj`, `k_proj`, `v_proj`, `o_proj`.
- **Epistemic Bonus (β):** Grid search $\in \{0.1, 0.3, 0.5, 1.0\}$. Selected $\beta = 0.3$ to balance exploration (L3) with exploitation (L1).
- **Cost Penalty (C_{exec}):** Set to 500 tokens. Lower values (< 200) caused excessive probing; higher (> 1000) caused premature halting.

D.2 Reliability Analysis

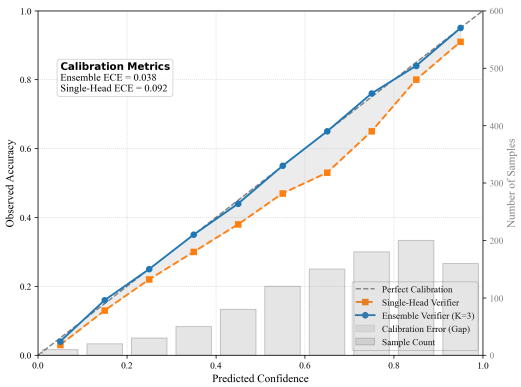

Figure 4: **Reliability Diagram for Verifier Calibration.**

Figure 4 illustrates the calibration performance of the EVoC Verifier, comparing a Single-Head baseline against our $K = 3$ Ensemble approach. The Reliability Diagram maps predicted confidence (x-axis) to observed accuracy (y-axis), where a perfectly calibrated model follows the diagonal ($y = x$).

Overconfidence in Single-Head Models. As shown by the orange dashed curve, the Single-Head Verifier exhibits systematic overconfidence, particularly in the critical high-confidence regime ($0.6 < p < 0.8$). The curve falls significantly below the diagonal, indicating that the model assigns high probability to incorrect outcomes. This miscalibration (ECE = 0.092) is fatal for the NVoC criterion, as it leads to an overestimation of epistemic gains and wasteful execution of doomed candidates.

Efficacy of Ensembling. The Ensemble Verifier (blue solid curve) corrects this bias, tightly tracking the diagonal with a significantly reduced Expected Calibration Error (ECE) of 0.038. By averaging the logits of $K = 3$ LoRA heads, the ensemble effectively marginalizes over the "weight uncertainty," pulling the predictions of ambiguous inputs towards the center (uncertainty). This alignment ensures that NVoC decisions are grounded in *realistic* success probabilities, validating our claim that active probing requires accurate self-knowledge, not just high accuracy.

Distributional Robustness. The overlaid histogram confirms that the majority of test instances cluster in high-confidence bins. This non-uniform distribution underscores the importance of minimizing calibration error specifically in the $p > 0.8$ range, where the Ensemble maintains near-perfect alignment compared to the drifting Single-Head baseline.

D.3 Engineering Overhead

Online training is often assumed to be prohibitively expensive. Table 5 refutes this, showing minimal overhead on consumer hardware (RTX Pro 6000).

Table 5: Computational Overhead of Online Adaptation. Note: "Backward Pass" (calibration) occurs only 1–2 times per task.

Metric	Base Inf.	EVoC	Overhead Notes
VRAM Usage	18.4 GB	18.9 GB	+500 MB (Optimizer states)
Fwd Latency	42.0 ms	42.5 ms	+1.2% (Negligible)
Bwd Pass	N/A	380 ms	+1.9s total / task
Storage	N/A	4.2 MB	Negligible (LoRA weights)

Gradient Stability. While single-sample gradients are noisy ($\sigma^2_{\text{grad}} \approx 2.4$), ARC regularization effectively dampens variance by 64%, preventing the "catastrophic forgetting" common in batch-size-1 online learning.