AgenTracer-v2: Agentic Failure Tracer for LLM Agentic Systems

Guibin Zhang[1,*], Haoyu Lu[2,*], Junhao Wang[3], He Zhu[4], Kun Wang[5]
Wangchunshu Zhou[6,†], Shuicheng Yan[1,†]
[1]NUS [2]NJU [3]CUHK [4]OPPO [5]NTU [6]ByteDance Inc.
[*]Equal contribution. [†]Corresponding authors.

Abstract

LLM-powered agentic systems increasingly operate across complex, long-horizon task regimes, yet their growing architectural complexity has intensified system-level fragility and failure rates. The task of *agentic failure attribution* seeks to identify the specific agent or execution step that causes failure within lengthy traces. However, most existing approaches remain constrained to passive, full-trajectory analysis, which limits their effectiveness in ultra-long and dynamically evolving trajectory attributions. To address this limitation, we introduce `AgenTracer-v2`, an autonomous, multi-hop failure attribution agent that reconceptualizes tracing as an active, tool-augmented diagnostic process. Concretely, we develop `Tracer-Flywheel`, a framework-agnostic data synthesis engine with deterministic system rollback, yielding an 8K dataset of precisely annotated failure trajectories. Leveraging this supervision, `AgenTracer-v2` is trained to perform token-efficient global summarization, targeted inward inspection, and outward exploration via integrated diagnostic tools. Extensive experiments demonstrate that `AgenTracer-v2` **(I)** consistently surpasses proprietary models such as GPT-5.2 and GEMINI-3-PRO, as well as prior specialized tracers, by up to 21.83% in attribution accuracy, **(II)** maintains robust performance on ultra-long trajectories exceeding 300K tokens and **(III)** delivering corrective feedback that measurably improves downstream agentic systems and multi-LLM reinforcement training. Codes are placed here.

1 Introduction

The rapid advancement of large language models (LLMs) has given rise to LLM-powered agentic systems that operate effectively in increasingly complex domains, including deep research [1, 2, 3], web browsing [4, 5, 6], and software engineering [7]. This progress stems not only from stronger agentic foundation models [8, 9], but also from the maturation of sophisticated agent scaffolds [10, 11, 10]. Such scaffolds commonly integrate short-/long-term memory systems [12, 13], multimodal perception modules, and comprehensive tool-use support (*e.g.*, MCPs [14] and agent skills [15]) that substantially expand the action space, which collectively endow agentic systems with greater flexibility and autonomy.

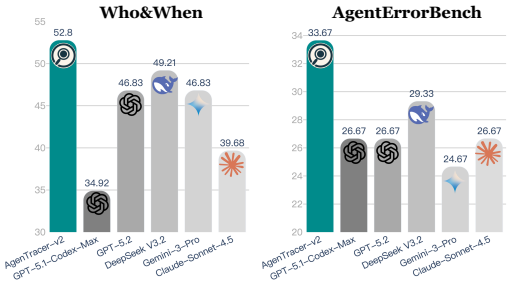

Figure 1: Failure attribution comparison between `AgenTracer-v2` and proprietary LLMs.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

However, this very degree of architectural sophistication also renders contemporary agentic systems increasingly **fragile at the system level**. During agent rollouts, tools may crash [16], memory modules may hallucinate or propagate incorrect states [17], planning components can adopt flawed objectives from the outset [18], and misalignment may arise between orchestrator agents and their sub-agents [19]. As long recognized in systems engineering, *complexity is the enemy of reliability* [20]. This vulnerability has been substantiated by empirical studies: popular frameworks such as OpenHands and MetaGPT exhibit failure rates of up to 86.7% [21], with failure modes spanning constraint ignorance, outcome misinterpretation, and others [22].

In response to this growing fragility, a rapidly expanding line of research has emerged around *agent failure attribution*, also referred to as *agent debugging* [23, 24, 25]. This line of work centers on the following question: given an agent execution trajectory that ultimately results in failure, how can one identify the *decisive error step* that causally drives the entire trajectory toward failure? Precisely localizing such critical errors is fundamental for enabling agentic systems to *correct*, *avoid*, and *learn from* their failures [22]. Early efforts on failure attribution for agentic systems primarily relied on manually designed workflows to assist off-the-shelf LLMs in improving error identification accuracy [26, 27, 28]. Recent studies have shifted toward internalizing failure-tracing capabilities through training, as seen in AgenTracer [23] and Aegis [25].

Despite these advances, existing failure tracers remain confined to the role of **passive readers**. Whether ingesting the full trajectory holistically [29, 23, 25], processing it sequentially [30, 22], or analyzing it hierarchically [31], these approaches invariably adhere to rigid scanning protocols. This static paradigm stands in sharp contrast to the cognitive process of real-world debugging: Human experts rarely consume logs linearly; instead, they dynamically skim for anomalies, drill down into suspicious content, cross-check intermediate states, and iteratively verify hypotheses. Bridging this gap necessitates a transition from passive readers to **proactive investigators**. We posit that an effective tracer must act as an autonomous agent, capable of deciding *what* to inspect and *how*, leveraging diagnostic tools on-the-fly, and engaging in multi-turn deliberation to pinpoint the root.

To instantiate this perspective, we introduce `AgenTracer-v2`, an autonomous, multi-hop failure attribution agent that operates via adaptive decision-making: internally, it selectively inspects long-horizon, multi-agent execution traces; externally, it performs online retrieval and code testing to ground diagnosis in real-world evidence. Naturally, training such an agent requires large-scale, high-fidelity supervision. We therefore develop `Tracer-Flywheel`, a framework-agnostic data synthesis engine with deterministic system rollback, enabling ♣ **counterfactual replay**, systematically replacing actions with oracle guidance to identify decisive error steps, and **simulated fault injection**, which introduces realistically distributed errors into successful trajectories to scale the dataset while preserving fidelity. Leveraging the resulting curated 8K dataset, we train `AgenTracer-v2` into a lightweight yet high-performing failure attributor via tool-integrated reinforcement learning. Our contributions can be summarized as follows:

❶ **Data Engine.** We introduce `Tracer-Flywheel`, a framework-agnostic automated data synthesis engine equipped with deterministic system rollback mechanism for scalable failure attribution, curating 8K high-fidelity failure trajectories across nine datasets.

❷ **Failure Tracer.** We train `AgenTracer-v2`, an autonomous failure-tracing agent that performs multi-hop, adaptive inspection over long-horizon agent trajectories and flexibly leverages diagnostic tools for inward inspection and outward exploration.

❸ **Empirical Evaluation.** Extensive experiments demonstrate that `AgenTracer-v2` achieves ***(I) accurate attribution***, outperforming GEMINI-3-PRO and GPT-5.2 by up to 21.83%; ***(II) stability***, maintaining high attribution accuracy on ultra-long trajectories exceeding 300K tokens; and ***(III) downstream utility***, improving the performance of off-the-shelf agentic systems by up to $\sim 21.2\%$ and boosting multi-LLM RL training by $\sim 4.3\%$.

2 Related Works

LLM-based Agent Systems. As foundation models with native agentic capabilities continue to advance [8, 32, 33], increasingly sophisticated agentic system scaffolds have emerged to amplify their effectiveness [34, 35, 36]. Contemporary LLM-based agent systems typically integrate a rich set of components, including planning modules for global decision-making [37, 38], memory systems that support both intra-episode working memory [39, 40] and cross-episode long-term memory [41, 42], tool integrations [14], as well as multi-agent communication protocols [43, 44, 45]. Yet this very

degree of compositional complexity also introduces substantial uncertainty and fragility. The tight coupling of heterogeneous modules significantly enlarges the space of failure modes, leading to alarmingly high system-level execution failure rates across a large volume of agent frameworks, as repeatedly documented in prior empirical studies [46, 21, 30].

Agent Failure Attribution. Developing specialized failure attributors has emerged as an effective strategy for mitigating the fragility of LLM agentic systems. Early studies revealed that off-the-shelf LLMs struggle to reliably attribute failures in long and complex execution traces [30], motivating subsequent work along two main directions: ■ **Workflow/scaffold-based approaches** introduce hand-crafted diagnostic procedures to assist existing LLMs in identifying errors more accurately, including binary search [30], hierarchical lookup [31], step-by-step inspection [22], and spectrum analysis [26]; ■ **Training-based approaches** aim to internalize failure attribution capabilities within the model itself, as exemplified by [23, 25]. Despite their differences, existing methods largely reduce failure attribution to single-pass consumption of entire trajectories, whereas `AgenTracer-v2` fundamentally departs from this by enabling proactive, autonomous, and multi-hop debugging. Additional related work discussion is provided in Appendix H.

3 Methodology

3.1 Problem Definition

We formulate agent system execution as a sequential decision process involving a set of agents $\mathcal{N} = \{1, \ldots, N\}$. At each discrete time step t, a single agent $\phi(t) \in \mathcal{N}$ is activated to produce an action $a_t \in \mathcal{A}$ based on the current system state. The system then evolves according to the transition function $s_{t+1} = \mathcal{T}(s_t, a_t)$. The execution trace is formalized as a trajectory $\xi = \langle s_0, a_0, s_1, a_1, \ldots, s_L \rangle$.

Agent Failure Attribution. Let $Z : \Xi \to \{0, 1\}$ denote the outcome verification function, where $Z(\xi) = 1$ indicates task success and $Z(\xi) = 0$ denotes failure. Following prior works [30, 23], we define agent failure attribution as identifying the *decisive error step*, namely the earliest step whose correction is sufficient to causally reverse the final outcome. We formalize this notion via a counterfactual intervention operator $\mathcal{I}_{\text{cf}}(\xi, t, a^*)$, which replaces the original action a_t with an oracle a^* at step t and re-simulates subsequent system dynamics. The decisive error step $t^\star$ and responsible agent $n^\star$ are defined as

$$\begin{aligned} t^\star &= \min \left\{ t \in [0, L] \mid \exists a^* \neq a_t : \; Z(\mathcal{I}_{\text{cf}}(\xi, t, a^*)) = 1 \right\}, \\ n^\star &= \phi(t^\star), \end{aligned} \tag{1}$$

which identifies the earliest point at which an intervention can steer the global execution from failure back to success. This definition serves as the ground-truth target for failure tracing.

3.2 Data Synthesis Engine: `Tracer-Flywheel`

Training robust failure attributors for multi-agent systems hinges on access to accurately annotated failure logs. Existing pipelines are either **manual**, high-effort and inherently small-scale (often $<$ 200 trajectories [22, 21, 30]), or **synthetic**, which injects perturbations into successful runs [29, 25] and may deviate from real failure distributions. A key bottleneck is operational: verifying a candidate decisive error at step t requires restoring the exact pre-action state s_t to enable controlled intervention, yet most agent frameworks provide no system-level rollback [47]. We therefore introduce `Tracer-Flywheel`, a lightweight, framework-agnostic data engine that enables deterministic rollback for scalable decisive-error annotation. Building on prior dual-engine designs, *i.e.*, counterfactual replay and fault injection in AgenTracer [23], `Tracer-Flywheel` emphasizes a faithful rollback mechanism to improve annotation precision and reliability. A more detailed elaboration between `Tracer-Flywheel` and AgenTracer is placed in Appendix A.

Decisive Error Annotation via Counterfactual Rollback. To extract decisive error labels from failed trajectories, we design an oracle-driven counterfactual annotation procedure enabled by a virtual rollback mechanism. Given a failed trajectory $\xi = \langle s_0, a_0, \ldots, s_L \rangle$ with $Z(\xi) = 0$, an oracle policy $\pi^\star$ sequentially inspects each step $t \in [0, L]$ and proposes a corrective action $a_t^\star \sim \pi^\star(s_t, \xi_{0:t})$ when the original action is deemed erroneous.

Since most agent frameworks lack native system-level rollback, we introduce a *virtual rollback operator* $\mathcal{R}(\xi, t)$, which deterministically replays the agentic system from the initial state while

enforcing state consistency with the original trajectory up to step t. The replayed state sequence $\{\tilde{s}_k\}$ is defined as

$$\tilde{s}_{k+1} = s_{k+1} \text{ if } k < t; \quad \mathcal{T}(\tilde{s}_t, a_t^\star) \text{ if } k = t; \quad \mathcal{T}(\tilde{s}_k, \tilde{a}_k) \text{ if } k > t. \tag{2}$$

where for $k < t$, all environment states, agent observations, tool states, and memories are synchronized with the original failed rollout, and for $k > t$ the system evolves freely under its native dynamics. The operator $\mathcal{I}_{\text{cf}}(\xi, t, a)$ first restores the pre-action state $\tilde{s}_t = \mathcal{R}(\xi, t)$, applies the replacement action a, and resumes execution while retaining the original trajectory prefix. The resulting counterfactual trajectory is

$$\tilde{\xi}^{(t)} = \mathcal{I}_{\text{cf}}(\xi, t, a_t^\star)\,. \tag{3}$$

If $Z(\tilde{\xi}^{(t)}) = 1$, the intervention is successful and t is identified as a valid decisive error candidate. Scanning from $t = 0$ onward and terminating at the earliest such step yields annotations consistent with Equation (1). Each failure instance is thus labeled as $(\xi, n^\star, t^\star)$, comprising the failed trajectory, the responsible agent, and the decisive error step.

Agentic Fault Injection. To further scale the dataset, we introduce an *injection agent* ψ that synthesizes realistic failures from originally successful executions. The injection agent is equipped with a repository of empirically distilled failure patterns $\mathcal{P}$, which directly adopts 17 agent error types distilled from real-world failure trajectories in AgentErrorBench [22], covering aspects such as memory, reflection, planning, and action. see Appendix B for full details. Given a successful trajectory $\xi = \langle s_0, a_0, \ldots, s_L \rangle$ with $Z(\xi) = 1$, the injection agent ψ selects an intervention step t and retrieves a context-compatible pattern $p \in \mathcal{P}$. Conditioned on (s_t, a_t, p), ψ generates a corrupted action $\tilde{a}_t \sim \psi(s_t, a_t, p)$, which is injected via the same virtual rollback operator $\mathcal{R}$. Specifically, we reconstruct the pre-intervention state using $\mathcal{R}(\xi, t)$ and intervene at step t, yielding

$$\tilde{\xi}^{(t)} = \mathcal{I}_{\text{cf}}(\xi, t, \tilde{a}_t)\,. \tag{4}$$

If the injected intervention induces failure, *i.e.*, $Z(\tilde{\xi}^{(t)}) = 0$, the resulting trajectory is retained as a valid synthetic failure instance and labeled as $(\tilde{\xi}^{(t)}, n^\star, t^\star)$, where $(n^\star, t^\star) = (\phi(t), t)$ is deterministically specified by construction.

Dataset Curation. We curate our dataset using five agentic systems spanning diverse domains and complexity levels. Single-agent frameworks include ReAct [48], OpenHands [10], and Flash-Searcher [38], while multi-agent frameworks comprise Cognitive-Kernel Pro (CK-Pro) [49] and AgentOrchestra [50]. These systems are evaluated across nine representative benchmarks: GAIA [2], TaskCraft [51], ToolQA [52], WebWalkerQA [53], xBench-DeepSearch [1], DeepSearchQA [54], AssistantBench [55], Birds [56], and KodCode [57]. For agent execution, we employ GPT-5-MINI and DEEPSEEK-V3.1 as the underlying language models. Both the oracle policy $\pi^\star$ and the injection agent ψ are instantiated using DEEPSEEK-V3.2.

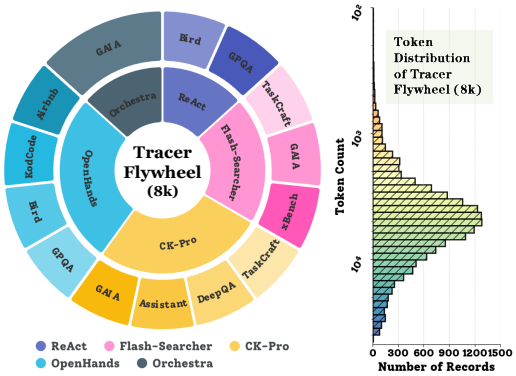

Figure 2: The data distribution curated by `Tracer-Flywheel`.

By running `Tracer-Flywheel` over the five agent frameworks and nine benchmarks described above, we curate a dataset $\mathcal{D} = \{(\xi_i, n_i^\star, t_i^\star)\}_{i=1}^{|\mathcal{D}|}$, consisting of $|\mathcal{D}| = 8{,}396$ precisely annotated failed trajectories, where each instance comprises a failed trajectory ξ_i, the responsible agent $n_i^\star$, and the decisive error step $t_i^\star$. The dataset is composed of two complementary subsets: $\mathcal{D}_{\text{cf}}$, obtained via counterfactual rollback, and $\mathcal{D}_{\text{inj}}$, synthesized through agentic fault injection, with $|\mathcal{D}_{\text{cf}}| + |\mathcal{D}_{\text{inj}}| = |\mathcal{D}|$. Trajectory lengths, as shown in Figure 2, span a wide range from 1 to 400K tokens, capturing agent executions of varying horizons and difficulty levels and providing rich and diverse supervision for training robust and scalable failure tracers. More dataset details are placed in Appendix C.

3.3 Agentic Attributor: `AgenTracer-v2`

With a sufficiently large and diverse dataset in place, we proceed to train `AgenTracer-v2`. Different from AgenTracer [23], which ingests an entire execution trajectory in a single pass and outputs

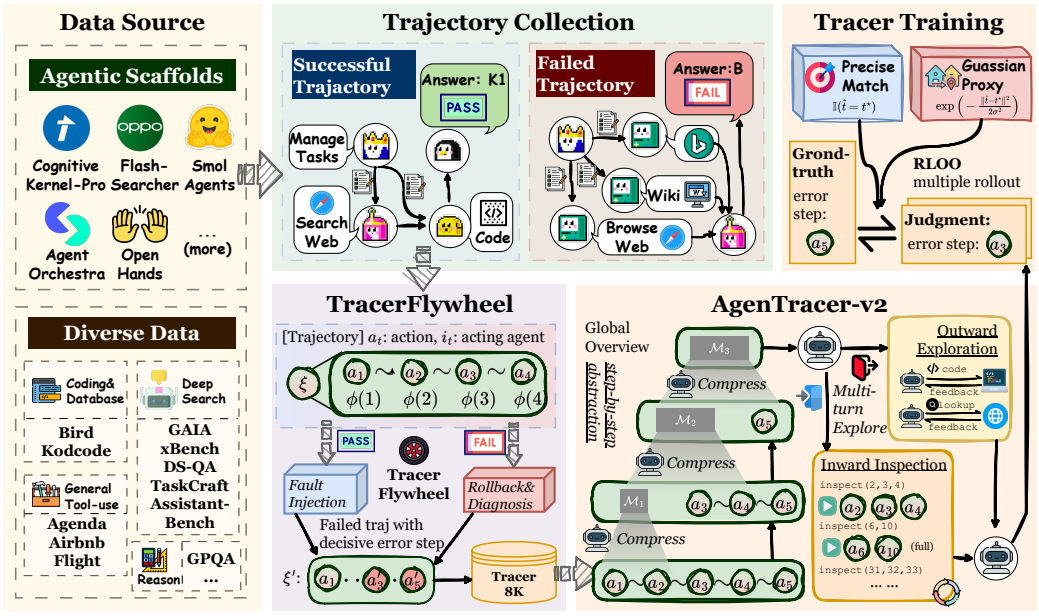

Figure 3: Overview of **AgenTracer-v2**. We first employ **Tracer-Flywheel** to collect and annotate execution trajectories, and subsequently train a multi-hop attributor agent capable of iterative, fine-grained failure diagnosis over long-horizon traces.

an attribution, **AgenTracer-v2** evolves into an autonomous attributor agent that performs adaptive, multi-hop decision-making over the trajectory. This design is motivated by two considerations. First, as discussed in Section 3.2, the collected trajectories extend to 400K tokens, far surpassing the context limits of many LLMs and rendering one-shot ingestion impractical. Second, moving beyond rigid, predefined debugging workflows more faithfully reflects real-world debugging practices. In what follows, we detail the design of **AgenTracer-v2**, whose core philosophy can be succinctly summarized as: *global overview*, *iterative inward inspection*, and *active outward exploration*. Detailed pseudo-algorithms are placed in Appendix F.

Global Overview. Given a failed trajectory $\xi = \langle s_0, a_0, \ldots, s_L \rangle$ with $Z(\xi) = 0$, the objective of the global overview stage is to allow **AgenTracer-v2** to rapidly acquire a holistic understanding of the execution in a token-efficient manner. **AgenTracer-v2** is first instantiated as a policy π_θ that incrementally consumes ξ in fixed-size chunks and maintains a persistent global diagnostic memory, rather than explicitly retaining the full trace. Let $\mathcal{M}_t$ denote the global diagnostic memory maintained by π_θ after processing steps up to time t. At iteration i, π_θ observes a window of d consecutive state-action pairs,

$$\xi_{[id:(i+1)d)} = \langle s_{id}, a_{id}, \ldots, s_{(i+1)d} \rangle, \tag{5}$$

and updates its memory according to

$$\mathcal{M}_{(i+1)d} \sim \pi_\theta\big(\mathcal{M}_{id} \mid \xi_{[id:(i+1)d)}, \mathcal{Q}\big), \tag{6}$$

where $\mathcal{Q}$ encodes the failure attribution query, including the task specification and the failure signal. Importantly, once this update is completed, the raw segment $\xi_{[id:(i+1)d)}$ is irrevocably discarded from the context window, ensuring that the memory footprint of π_θ remains independent of the trajectory length. After iterating over the entire execution, the resulting memory state can serve as a compact, trajectory-level abstraction for subsequent reference:

$$\mathcal{M}_L \sim \pi_\theta\big(\mathcal{M}_0 \mid \{\xi_{[0:d)}, \xi_{[d:2d)}, \ldots, \xi_{[L-d:L)}\}, \mathcal{Q}\big). \tag{7}$$

Iterative Inward Inspection. Conditioned on the global summary $\mathcal{M}_L$ and query $\mathcal{Q}$, **AgenTracer-v2** performs fine-grained diagnosis by iteratively probing selected steps of the trajectory via a diagnostic tool `InspectStep`. At round τ, π_θ selects a set of step indices to inspect,

$$\mathcal{T}_\tau \sim \pi_\theta(\mathcal{T} \mid \mathcal{M}_L, \mathcal{Q}, h_{\tau-1}), \ \mathcal{T}_\tau \subseteq [0, L], \tag{8}$$

where $h_{\tau-1}$ denotes the accumulated interaction history. Executing $\texttt{InspectStep}(\mathcal{T}_\tau)$ returns a collection of localized observations $\mathcal{O}_\tau = \{\texttt{InspectStep}(t) \mid t \in \mathcal{T}_\tau\}$, each containing detailed

state, action, tool I/O, and intermediate artifacts associated with the corresponding steps. By repeating this process, `AgenTracer-v2` progressively refines its hypothesis through targeted inspection, non-sequentially traversing related steps to construct coherent causal chains underlying the failure.

Active Outward Exploration. While inward inspection enables detailed analysis of execution traces, failure diagnosis based solely on internal reasoning may drift without grounding in real-world evidence. To anchor attribution in external reality, we equip `AgenTracer-v2` with two auxiliary tools, `OnlineSearch` and `CodingTest`, instantiated under the *agent-as-tool* paradigm [58, 59]. Each tool invokes an independent agent (disjoint in context from `AgenTracer-v2`) augmented with domain-specific capabilities, including web retrieval (*e.g.*, search engines and Jina crawlers) and code execution (*e.g.*, E2B sanboxes). At exploration step τ, `AgenTracer-v2` issues a verification query q_τ to an external tool $\mathcal{T} \in \{\texttt{OnlineSearch}, \texttt{CodingTest}\}$, which autonomously performs multi-step interaction with the environment and returns an evidence signal $e_\tau \sim \mathcal{T}(q_\tau)$, which informs subsequent reasoning, enabling `AgenTracer-v2` to validate intermediate claims, test executable artifacts, and substantiate diagnostic hypotheses with verifiable external signals.

3.4 Training Setup

We formulate the training of `AgenTracer-v2` as a policy optimization problem aimed at maximizing failure attribution accuracy. Formally, given a dataset of annotated failure instances $\mathcal{D} = \{(\xi, n^\star, t^\star)\}$, the attributor agent π_θ generates a diagnostic trajectory $\varpi = \langle q_0, o_0, q_1, o_1, \ldots, \hat{y}\rangle$, where q_k represents reasoning thoughts or tool invocations (*e.g.*, `InspectStep`), o_k denotes the corresponding environmental observations, and $\hat{y} = (\hat{n}, \hat{t})$ is the final attribution outcome. To effectively optimize π_θ in the presence of non-differentiable tool interactions and discrete reasoning steps, we adopt RLOO [60], a low-variance policy gradient estimator. For each input ξ, we sample K independent diagnostic trajectories $\{\varpi_1, \ldots, \varpi_K\}$ from the current policy $\pi_\theta(\cdot|\xi)$. The objective is defined as minimizing the expected loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\xi\sim\mathcal{D}}\left[\frac{1}{K}\sum_{k=1}^{K}\hat{A}(\varpi_k)\sum_{j=1}^{|\varpi_k|}\log\pi_\theta(q_{k,j} \mid \xi, \varpi_{k,<j})\right], \tag{9}$$

where $\hat{A}(\varpi_k)$ serves as the advantage estimate for the k-th trajectory, which is calculated by the mean reward of the peer samples without requiring an auxiliary value network:

$$\hat{A}(\varpi_k) = R(\varpi_k) - \frac{1}{K-1}\sum_{j\neq k} R(\varpi_j). \tag{10}$$

Reward Function. To navigate the sparse reward landscape of long-horizon failure attribution, we design a composite reward function $R(\varpi)$ that integrates structural constraints with both dense shaping signals and sparse completion signals. The reward is formalized as:

$$R(\varpi) = \mathbb{I}_{\text{fmt}}(\varpi)\cdot\Big(\underbrace{\exp\big(-\frac{\|\hat{t}-t^\star\|^2}{2\sigma^2}\big)}_{r_{\text{prox}}(\hat{t},t^\star)} + \gamma\cdot\underbrace{\mathbb{I}(\hat{t}=t^\star)}_{r_{\text{exact}}(\hat{t},t^\star)}\Big), \tag{11}$$

where $\mathbb{I}_{\text{fmt}}(\varpi) \in \{0, 1\}$ serves as a hard gating function, yielding non-zero returns solely when the output adheres to the requisite parsing format. The term r_{prox} provides a soft, dense supervision signal via a Gaussian kernel, incentivizing the predicted error step $\hat{t}$ to converge towards the ground truth $t^\star$ even when the precise attribution is missed. Finally, r_{exact} acts as a sparse perfection reward, granting a bonus $\gamma = 2$ if and only if the agent correctly identifies the decisive step, thereby aligning the optimization landscape with the ultimate attribution objective.

4 Experiments

4.1 Experiment Setup

Training Setup. We use QWEN3-4B-INSTRUCT [61] as the base model and train it using RLOO on the 8K annotated dataset collected in Section 3.2. All experiments are conducted on a single server equipped with eight NVIDIA H20 (96GB) GPUs. For RL training in Section 3.4, we employ

Table 1: Performance comparison on the Who&When/AgentErrorBench/TRAIL benchmarks. We report both agent-level and step-level accuracy. The best and second-best results are highlighted in **bold** and underlined, respectively. Methods with † are not open-sourced, so we directly adopt results from original papers; entries with "–" indicate that the method was not evaluated on the respective subset. Since both AgentErrorBench and TRAIL are single-agent trajectories, the agent-level accuracy is trivially saturated and provides limited discriminative value; accordingly, we render these entries in gray for clarity. We report the average of **three random runs**.

Model	Who&When Benchmark				AgentErrorBench						TRAIL	
	Hand-crafted		Algorithm-gen		GAIA		ALFWorld		Webshop		GAIA	
	Agent	Step	Agent	Step	Agent	Step	Agent	Step	Agent	Step	Agent	Step
Vanilla LLMs												
QWEN3-8B	63.79	18.97	53.97	30.16	100.00	22.00	100.00	2.33	100.00	12.00	100.00	0.72
QWEN3-32B	43.10	20.69	54.76	41.27	98.00	26.67	100.00	9.33	100.00	12.00	100.00	23.43
QWEN3-CODER	50.00	15.52	62.70	47.62	94.00	32.67	99.00	12.67	94.00	18.67	100.00	11.09
GPT-5.1-CODEX-MAX	43.10	24.14	59.52	34.92	100.00	26.67	100.00	17.67	100.00	22.00	100.00	37.82
GPT-5.2	34.48	17.24	65.08	46.83	100.00	26.67	100.00	17.67	100.00	26.00	100.00	24.19
DEEPSEEK-V3.2	39.66	18.97	63.49	49.21	98.00	29.33	99.33	10.33	100.00	10.00	100.00	19.76
GEMINI-3-PRO	48.28	20.69	65.87	46.83	98.67	24.67	99.50	15.50	100.00	24.00	100.00	45.13
CLAUDE-SONNET-4.5	48.28	25.86	59.52	39.68	100.00	26.67	100.00	10.00	100.00	17.33	100.00	32.53
Scaffold-based Methods												
CORRECT (+G-2.5-PRO)†	-	5.2	-	24.6	-	-	-	-	-	-	-	-
CORRECT (+GPT-5-NANO)†	-	6.9	-	24.6	-	-	-	-	-	-	-	-
CORRECT (+GPT-5)†	-	17.2	-	38.1	-	-	-	-	-	-	-	-
CasualMAS†	56.8	18.2	48.5	36.2	-	-	-	-	-	-	-	-
Training-based Methods												
AgenTracer	63.82	20.68	63.73	37.30	100.00	27.33	100.00	13.67	100.00	19.33	100.00	38.93
`AgenTracer-v2`	**70.19**	**28.17**	**74.17**	**52.80**	100.00	**33.67**	100.00	**20.33**	100.00	**28.33**	100.00	**46.02**

the rLLM[1] and verl[2] frameworks. We use a batch size of 32, a rollout number of 8, learning rate of 1×10^{-6}. The chunk size d in Equation (5) is 3. More training details are in Appendix D.

Benchmarks. We adopt three established agent failure attribution benchmarks: **Who&When** [30], which comprises *hand-crafted* and *algorithm-generated* subsets; **AgentErrorBench** [22], partitioned into *GAIA*, *ALFWorld*, and *WebShop* subsets based on trajectory source; and **TRAIL** [46]. Who&When and AgentErrorBench adopt the same protocol that labels each failed trajectory with the earliest decisive error step. TRAIL annotates multiple errors per trajectory; for consistency, we use the earliest annotated error as the label.

Evaluation Metrics. Following [30], we adopt two evaluation metrics: *agent-level accuracy* and *step-level accuracy*. Agent-level accuracy measures whether the attributor correctly identifies the responsible agent $n^\star$ for a failed trajectory, while step-level accuracy evaluates whether the decisive error step $t^\star$ is localized.

Baselines. We compare against three categories of baselines. ■ **Vanilla LLMs**: LLMs prompted under the all-at-once paradigm introduced in [30], including QWEN3-4B-INSTRUCT-2507, QWEN3-8B-THINKING [61], QWEN3-32B, QWEN3-CODER-480B-A35B-INSTRUCT (QWEN3-CODER) [61], GPT-5.1-CODEX, GPT-5.2 [62], GEMINI-3-PRO [63], CLAUDE-SONNET-4.5 [64], DEEPSEEK-V3.2 [65]; ■ **Scaffold-based methods**: CORRECT [29] and CausalMAS [66]; ■ **Training methods**: AgenTracer [23].

4.2 Application I: Failure Attribution

A primary criterion for assessing a failure tracer is the accuracy of its failure attribution. In Table 1, we compare **`AgenTracer-v2`** against 15 baselines on Who&When, AgentErrorBench, and TRAIL benchmarks, reporting both agent- and step-level accuracy. We observe:

`AgenTracer-v2` outperforms giant Models in failure attribution. As shown in Table 1, model scale alone does not guarantee reliable attribution performance. Even leading proprietary models such as GEMINI-3-PRO and CLAUDE-SONNET-4.5 achieve step-level accuracies below 18% on AgentErrorBench-ALFWorld, underscoring the necessity of specialized failure tracers. Likewise, scaffold-based approaches equipped with GPT-5 remain limited (e.g., 36.2% on Who&When-Algo).

[1] https://github.com/rllm-org/rllm
[2] https://github.com/volcengine/verl

In contrast, **AgenTracer-v2**, despite its lightweight 4B parameterization, consistently outperforms these baselines, exceeding GPT-5.2 by 10.94% on Who&When-Hand and surpassing GEMINI-3-PRO by 9.0% on AgentErrorBench-GAIA. It also improves over trained counterparts such as AgenTracer (e.g., +7.5% on Who&When-Hand). These results highlight the effectiveness and necessity of a dedicated, agentic failure tracer.

AgenTracer-v2 excels at long-horizon trajectory attribution. A defining feature of **AgenTracer-v2** is its tailored treatment of long-context trajectories (*i.e.*, global summarization). We compare **AgenTracer-v2** with representative baselines on trajectories exceeding 10K, 100K, and 300K tokens (Figure 4). Although large-scale models such as GPT-5.2 and GEMINI-3-PRO support extended contexts, their attribution accuracy deteriorates sharply with length (e.g., GPT-5.2 falls from 23.0% on full trajectories to 2.3% at $>$ 300K, a 20.7% drop). In contrast, **AgenTracer-v2** degrades more gracefully and retains a clear advantage in the $>$ 300K regime, surpassing GPT-5.2 by 17.0% and GEMINI-3-PRO by 14.5%, underscoring its robustness for long-horizon failure attribution.

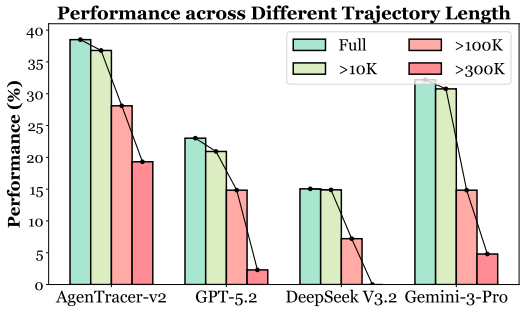

Figure 4: Step-level attribution accuracy as a function of trajectory length, comparing **AgenTracer-v2** with representative baselines across increasing context regimes.

4.3 Application II: Agent System Self-Improvement

We further assess the practical utility of **AgenTracer-v2** by evaluating whether tracer-provided feedback improves the performance of off-the-shelf agentic systems. Following [23], we compare against Self-Refine [67] and CRITIC [68] (both instantiated with DEEPSEEK-V3.2), as well as AgenTracer. For each failed trajectory ξ with $Z(\xi) = 0$, the tracers feedback is provided to the agentic system to trigger re-execution, leveraging external critique to enhance performance. This procedure is iterated for four rounds.

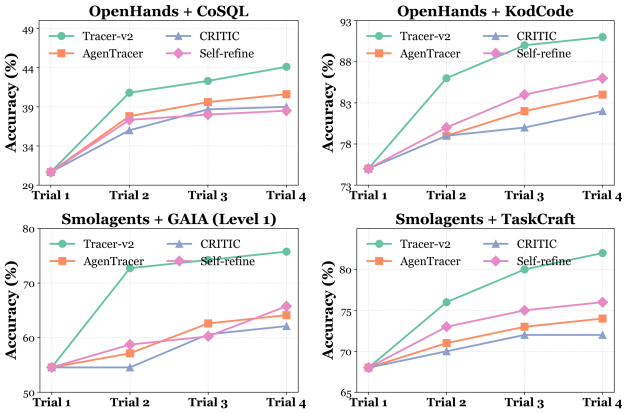

Figure 5: Multi-round performance gains of OpenHands and Smolagents guided by tracer feedback, comparing **AgenTracer-v2** with baseline tracers.

AgenTracer-v2 Facilitates Agent Self-Debugging and Self-Improvement. In Figure 5, we adopt GLM-4.5 as the base model for OpenHands and Smolagents and conduct multi-round, tracer-guided problem solving across CoSQL [69], KodCode, GAIA (Level 1), and TaskCraft. The TaskCraft and CoSQL test sets are fully held out from training, ensuring that the reported gains reflect genuine generalization. **AgenTracer-v2** exhibits consistent improvements from trial-1 to trial-4 on all benchmarks, including +13.4% on OpenHands+CoSQL and +21.2% on Smolagents+GAIA. As shown in Figure 5, it substantially outperforms AgenTracer, achieving final accuracy gains of +11.6% on GAIA and +8% on TaskCraft, and consistently surpasses Self-Refine/CRITIC baselines powered by DEEPSEEK-V3.2, demonstrating reliable, actionable corrective feedback across diverse domains and systems.

4.4 Application III: Tracer as Reward Model

We further evaluate the downstream utility of `AgenTracer-v2` by integrating it into existing multi-LLM-agent reinforcement learning frameworks. Long-horizon RL in agentic systems is often hindered by sparse terminal rewards and imbalanced credit assignment across extended rollouts [70]. In this setting, a failure tracer can serve as a principled mechanism for fine-grained credit attribution along long execution traces. Concretely, during online RL, when a rollout terminates in failure, we invoke `AgenTracer-v2` to diagnose the responsible agent, and assign an additional process-level negative reward to the agents' associated steps. We embed this mechanism into the popular MARTI framework [71] and observe consistent $\sim 4.28\%$ performance gains on both coding and mathematical tasks, as shown in Figure 6. For more detailed setup, refer to Appendix D.2.

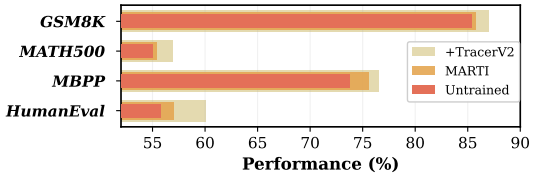

Figure 6: Integration of `AgenTracer-v2` into MARTI, illustrating tracer-guided credit assignment and RL performance gains using QWEN2.5-3B-INSTRUCT as the base model.

Table 2: **Ablation study** of `AgenTracer-v2` components, evaluated on AgentErrorBench (GAIA) and Who&When (algo).

Method	AgentError		Who&When-algo	
	Agent	Step	Agent	Step
`AgenTracer-v2`	100	33.7	74.2	52.8
w/o agentic	100	28.0	68.2	42.1
w/o external	100	32.0	69.8	49.2
w/o internal	100	28.0	67.5	44.4

Table 3: **Efficiency analysis** of `AgenTracer-v2` on the Who&When-algo dataset, reporting the average step-level accuracy, token consumption, and wall-clock time.

Method / Component	Step Acc. (%)	Tokens (K)	Avg. Wall Time (s)
GPT-5.2	46.83	38.7	31.4
Gemini-3-Pro	46.83	41.2	34.8
AgenTracer	37.30	29.5	16.2
`AgenTracer-v2` (total)	**52.80**	53.6	42.7
— Global Summarization	–	22.4	14.3
— Inward Inspection	–	24.8	19.6
— Outward Tool Calls	–	6.4	8.8

4.5 Ablation Study

To assess the contribution of each component in `AgenTracer-v2`, we evaluate three variants: *w/o agentic*, which disables multi-hop autonomous querying and reverts to a single-pass, AgenTracer-style diagnosis; *w/o external*, which removes access to external diagnostic tools; and *w/o internal*, which disables iterative step inspection via `InspectStep`. As shown in Table 2, `AgenTracer-v2` achieves the highest step-level accuracy on both benchmarks. Removing the agentic multi-hop mechanism (*w/o agentic*) produces the largest drop (-5.7% on AgentError, -10.7% on Who&When-algo), underscoring the importance of adaptive, multi-turn reasoning for precise localization. Excluding external tools (*w/o external*) results in a moderate decline, while disabling internal step inspection (*w/o internal*) similarly degrades performance, confirming the complementary roles of external verification and fine-grained trace probing.

4.6 Efficiency Analysis

As shown in Table 3, `AgenTracer-v2` requires more tokens and wall-clock time than single-pass proprietary baselines and the previous AgenTracer variant, averaging 53.6K tokens ($< 1.5\times$) and 42.7s ($< 2\times$) per trajectory. This overhead is moderate, since reasoning-heavy models such as GPT-5.2 also spend substantial reasoning tokens on single-pass trajectory ingestion, while `AgenTracer-v2` reduces unnecessary computation via concise compression and targeted inspection. The cost is justified because failure attribution is typically an offline diagnostic task where accuracy matters more than latency. Moreover, `AgenTracer-v2` improves over the strongest baseline by $+5.97\%$ with only about $1.30\times$ the token usage of Gemini-3-Pro, placing it on the accuracy–efficiency Pareto frontier. As a compact 4B-parameter model, it is also much cheaper to deploy than proprietary large-scale models, making the additional agentic inspection cost practically acceptable.

4.7 Case Study

Figure 7 illustrates how `AgenTracer-v2` performs proactive, multi-hop diagnosis on a real failure case. Starting from a compressed global summary, the agent first localizes suspicious regions (steps

2830) via targeted inspection, then invokes `OnlineSearch` to verify ambiguous evidence. Upon cross-checking earlier relevant steps (17 and 21), **`AgenTracer-v2`** identifies a causal inconsistency: the agent prematurely abandoned evidence-based search and fabricated a unverified assumption at step 30. This example demonstrates how **`AgenTracer-v2`** constructs a grounded causal chain via adaptive inspection rather than relying on passive, single-pass trace reading.

5 Conclusion

We introduce **`AgenTracer-v2`**, an autonomous, multi-hop failure tracer that performs adaptive inspection and tool-grounded verification, together with **`Tracer-Flywheel`**, a framework-agnostic data engine that enables large-scale data synthesis of annotated failure trajectories. Extensive evaluation across diverse benchmarks demonstrates that **`AgenTracer-v2`** achieves consistent gains in attribution accuracy over large proprietary models and prior specialized tracers. Beyond attribution, **`AgenTracer-v2`** provides actionable feedback that improves downstream agentic systems and serves as a principled credit-assignment signal for long-horizon multi-agent reinforcement learning. Together, these results establish a scalable, general, and practically deployable foundation for proactive, self-improving diagnosis in next-generation agentic AI.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 62320106007, and by NUS Grant A-0010106-00-00, A-8004365-00-00 and A-8004410-01-00.

References

[1] Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, Kenkun Liu, Rui Wang, Run Li, Tong Niu, Wenlong Zhang, Wenqi Yan, Xuanzheng Wang, Yuchen Zhang, Yi-Hsin Hung, Yuan Jiang, Zexuan Liu, Zihan Yin, Zijian Ma, and Zhiwen Mo. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations, 2025.

[2] Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

[3] Zhengliang Shi, Yiqun Chen, Haitao Li, Weiwei Sun, Shiyu Ni, Yougang Lyu, Run-Ze Fan, Bowen Jin, Yixuan Weng, Minjun Zhu, Qiujie Xie, Xinyu Guo, Qu Yang, Jiayi Wu, Jujia Zhao, Xiaqiang Tang, Xinbei Ma, Cunxiang Wang, Jiaxin Mao, Qingyao Ai, Jen-Tse Huang, Wenxuan Wang, Yue Zhang, Yiming Yang, Zhaopeng Tu, and Zhaochun Ren. Deep research: A systematic survey, 2025.

[4] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025.

[5] Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding?, 2024.

[6] Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Jihyung Kil, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. GUI agents: A survey. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538, Vienna, Austria, July 2025. Association for Computational Linguistics.

[7] Huanting Wang, Jingzhi Gong, Huawei Zhang, Jie Xu, and Zheng Wang. Ai agentic programming: A survey of techniques, challenges, and opportunities, 2025.

[8] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025.

[9] Chen Hu, Haikuo Du, Heng Wang, Lin Lin, Mingrui Chen, Peng Liu, Ruihang Miao, Tianchi Yue, Wang You, Wei Ji, Wei Yuan, Wenjin Deng, Xiaojian Yuan, Xiaoyun Zhang, Xiangyu Liu, Xikai Liu, Yanming Xu, Yicheng Cao, Yifei Zhang, Yongyao Wang, Yubo Shu, Yurong Zhang, Yuxiang Zhang, Zheng Gong, Zhichao Chang, Binyan Li, Dan Ma, Furong Jia, Hongyuan Wang, Jiayu Liu, Jing Bai, Junlan Liu, Manjiao Liu, Na Wang, Qiuping Wu, Qinxin Du, Shiwei Li, Wen Sun, Yifeng Gong, Yonglin Chen, Yuling Zhao, Yuxuan Lin, Ziqi Ren, Zixuan Wang, Aihu Zhang, Brian Li, Buyun Ma, Kang An, Li Xie, Mingliang Li, Pan Li, Shidong Yang, Xi Chen, Xiaojia Liu, Yuchu Luo, Yuan Song, YuanHao Ding, Yuanwei Liang, Zexi Li, Zhaoning Zhang, Zixin Zhang, Binxing Jiao, Daxin Jiang, Jiansheng Chen, Jing Li, Xiangyu Zhang, and Yibo Zhu. Step-deepresearch technical report, 2025.

[10] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents, 2025.

[11] LangGraph. GitHub - langchain-ai/langgraph: Build resilient language agents as graphs. — github.com. https://github.com/langchain-ai/langgraph, 2025. [Accessed 03-01-2026].

[12] Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of ai agents, 2025.

[13] Zeyu Zhang, Quanyu Dai, Xu Chen, Rui Li, Zhongyang Li, and Zhenhua Dong. Memengine: A unified and modular library for developing advanced memory of llm-based agents, 2025.

[14] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model context protocol (mcp): Landscape, security threats, and future research directions, 2025.

[15] ClaudeCode. Agent Skills - Claude Code Docs — code.claude.com. https://code.claude.com/docs/en/skills, 2025. [Accessed 03-01-2026].

[16] Shirley Kokane, Ming Zhu, Tulika Awalgaonkar, Jianguo Zhang, Thai Hoang, Akshara Prabhakar, Zuxin Liu, Tian Lan, Liangwei Yang, Juntao Tan, Rithesh Murthy, Weiran Yao, Zhiwei Liu, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, Caiming Xiong, and Silivo Savarese. Toolscan: A benchmark for characterizing errors in tool-use llms, 2025.

[17] Ding Chen, Simin Niu, Kehang Li, Peng Liu, Xiangping Zheng, Bo Tang, Xinchi Li, Feiyu Xiong, and Zhiyu Li. Halumem: Evaluating hallucinations in memory systems of agents, 2025.

[18] Zhenlan Ji, Daoyuan Wu, Pingchuan Ma, Zongjie Li, and Shuai Wang. Testing and understanding erroneous planning in llm agents through synthesized user inputs, 2024.

[19] Florian Carichon, Aditi Khandelwal, Marylou Fauchard, and Golnoosh Farnadi. The coming crisis of multi-agent misalignment: Ai alignment must be a dynamic and social process, 2025.

[20] Marvin Rausand and Arnljot Hoyland. *System reliability theory: models, statistical methods, and applications*, volume 396. John Wiley & Sons, 2003.

[21] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[22] Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, Xiaoteng Ma, Xiaodong Yu, Gowtham Ramesh, Jialian Wu, Zicheng Liu, Pan Lu, James Zou, and Jiaxuan You. Where llm agents fail and how they can learn from failures, 2025.

[23] Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. Agentracer: Who is inducing failure in the llm agentic systems?, 2025.

[24] Kevin Song, Anand Jayarajan, Yaoyao Ding, Qidong Su, Zhanda Zhu, Sihang Liu, and Gennady Pekhimenko. Aegis: Taxonomy and optimizations for overcoming agent-environment failures in llm agents, 2025.

[25] Fanqi Kong, Ruijie Zhang, Huaxiao Yin, Guibin Zhang, Xiaofei Zhang, Ziang Chen, Zhaowei Zhang, Xiaoyuan Zhang, Song-Chun Zhu, and Xue Feng. Aegis: Automated error generation and attribution for multi-agent systems, 2025.

[26] Yu Ge, Linna Xie, Zhong Li, Yu Pei, and Tian Zhang. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. *arXiv preprint arXiv:2509.13782*, 2025.

[27] Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. Deepscientist: Advancing frontier-pushing scientific findings progressively, 2025.

[28] Heng Zhang, Yuling Shi, Xiaodong Gu, Haochen You, Zijian Zhang, Lubin Gan, Yilei Yuan, and Jin Huang. Graphtracer: Graph-guided failure tracing in llm agents for robust multi-turn deep search. *arXiv preprint arXiv:2510.10581*, 2025.

[29] Yifan Yu, Moyan Li, Shaoyuan Xu, Jinmiao Fu, Xinhai Hou, Fan Lai, and Bryan Wang. Correct: Condensed error recognition via knowledge transfer in multi-agent systems. *arXiv preprint arXiv:2509.24088*, 2025.

[30] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025.

[31] Adi Banerjee, Anirudh Nair, and Tarik Borogovac. Where did it all go wrong? a hierarchical look into multi-agent error attribution. *arXiv preprint arXiv:2510.04886*, 2025.

[32] 5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025.

[33] MiniMax. MiniMax M2 — minimax.io. https://www.minimax.io/news/minimax-m2, 2025. [Accessed 04-01-2026].

[34] Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-swe-agent: Can software engineering agents self-evolve on the fly?, 2025.

[35] MiniMax. GitHub - MiniMax-AI/Mini-Agent: A minimal yet professional single agent demo project that showcases the core execution pipeline and production-grade features of agents. — github.com. https://github.com/MiniMax-AI/Mini-Agent, 2025. [Accessed 04-01-2026].

[36] Anthropics. GitHub - anthropics/claude-code: Claude Code is an agentic coding tool that lives in your terminal, understands your codebase, and helps you code faster by executing routine tasks, explaining complex code, and handling git workflows - all through natural language commands. — github.com. https://github.com/anthropics/claude-code, 2025. [Accessed 04-01-2026].

[37] He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, Jian Yang, Minghao Liu, Xitong Gao, Jiaheng Liu, and Wangchunshu Zhou. Oagents: An empirical study of building effective agents, 2025.

[38] Tianrui Qin, Qianben Chen, Sinuo Wang, He Xing, King Zhu, He Zhu, Dingfeng Shi, Xinxin Liu, Ge Zhang, Jiaheng Liu, Yuchen Eleanor Jiang, Xitong Gao, and Wangchunshu Zhou. Flash-searcher: Fast and effective web agents via dag-based parallel execution, 2025.

[39] Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding, 2025.

[40] Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks, 2025.

[41] Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning, 2025.

[42] Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025.

[43] Samuele Marro, Emanuele La Malfa, Jesse Wright, Guohao Li, Nigel Shadbolt, Michael Wooldridge, and Philip Torr. A scalable communication protocol for networks of large language models, 2024.

[44] Google. Announcing the Agent2Agent Protocol (A2A) — developers.googleblog.com. https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/, 2025. [Accessed 04-01-2026].

[45] Gaowei Chang, Eidan Lin, Chengxuan Yuan, Rizhao Cai, Binbin Chen, Xuan Xie, and Yin Zhang. Agent network protocol technical white paper, 2025.

[46] Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. Trail: Trace reasoning and agentic issue localization, 2025.

[47] Yang Li, Siqi Ping, Xiyu Chen, Xiaojian Qi, Zigan Wang, Ye Luo, and Xiaowei Zhang. Agentgit: A version control framework for reliable and scalable llm-powered multi-agent systems, 2025.

[48] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[49] Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, Hongming Zhang, Haitao Mi, and Dong Yu. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training, 2025.

[50] Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating hierarchical multi-agent intelligence with the tool-environment-agent(tea) protocol, 2025.

[51] Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Taskcraft: Automated generation of agentic tasks, 2025.

[52] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools, 2023.

[53] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. Webwalker: Benchmarking llms in web traversal, 2025.

[54] Google. DeepSearchQA — kaggle.com. https://www.kaggle.com/datasets/deepmind/deepsearchqa, 2025. [Accessed 05-01-2026].

[55] Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. Assistantbench: Can web agents solve realistic and time-consuming tasks?, 2024.

[56] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023.

[57] Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding, 2025.

[58] Yanfei Zhang. Agent-as-tool: A study on the hierarchical decision making with reinforcement learning, 2025.

[59] Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Yang Zhao, Hongjin Qian, and Zhicheng Dou. Hira: A hierarchical reasoning framework for decoupled planning and execution in deep search, 2025.

[60] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024.

[61] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

[62] OpenAI. Introducing GPT-5.2 — openai.com. https://openai.com/index/introducing-gpt-5-2/, 2025. [Accessed 08-01-2026].

[63] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[64] Anthropic. Claude Sonnet 4. https://www.anthropic.com/claude/sonnet, 2025. [Accessed 31-08-2025].

[65] DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li,

Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025.

[66] Guoqing Ma, Jia Zhu, Hanghui Guo, Weijie Shi, Jiawei Shen, Jingjiang Liu, and Yidan Liang. Automatic failure attribution and critical step prediction method for multi-agent systems based on causal inference. *arXiv preprint arXiv:2509.08682*, 2025.

[67] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

[68] Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024.

[69] Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S Lasecki, and Dragomir Radev. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases, 2019.

[70] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025.

[71] MARTI. GitHub - TsinghuaC3I/MARTI: A Framework for LLM-based Multi-Agent Reinforced Training and Inference — github.com. https://github.com/{T}singhua{C}3{I}/{M}{A}{R}{T}{I}, 2025.

[72] Dayuan Fu, Jianzhao Huang, Siyuan Lu, Guanting Dong, Yejie Wang, Keqing He, and Weiran Xu. Preact: Prediction enhances agent's planning ability. *arXiv preprint arXiv:2402.11534*, 2024.

[73] Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving, 2025.

[74] Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

[75] Gaole Dai, Shiqi Jiang, Ting Cao, Yuqing Yang, Yuanchun Li, Rui Tan, Mo Li, and Lili Qiu. ProRe: A proactive reward system for GUI agents via reasoner-actor collaboration. In *The Fourteenth International Conference on Learning Representations*, 2026.

[76] Zhisong Qiu, Shuofei Qiao, Kewei Xu, Yuqi Zhu, Lun Du, Ningyu Zhang, and Huajun Chen. Rewarding the scientific process: Process-level reward modeling for agentic data analysis. *arXiv preprint arXiv:2604.24198*, 2026.

[77] Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

[78] Yunpeng Dong, Jingkai He, Shiqi Liu, Yuze Hou, Dong Du, Zhonghu Xu, Si Yu, Baochuan Yang, Yubin Xia, and Haibo Chen. DeltaBox: Scaling stateful AI agents with millisecond-level sandbox checkpoint/rollback. *arXiv preprint arXiv:2605.22781*, 2026.

[79] Tianyuan Wu, Chaokun Chang, Lunxi Cao, Wei Gao, and Wei Wang. Crab: A semantics-aware checkpoint/restore runtime for agent sandboxes. *arXiv preprint arXiv:2604.28138*, 2026.

[80] Yusheng Zheng, Yiwei Yang, Wei Zhang, and Andi Quinn. ACRFence: Preventing semantic rollback attacks in agent checkpoint-restore. *arXiv preprint arXiv:2603.20625*, 2026.

A Data Engine

Design Objective. The core objective of `Tracer-Flywheel` is to enable *faithful, system-level counterfactual intervention* in heterogeneous, open-ended agentic environments. Formally, given a failed execution trajectory $\xi = \langle s_0, a_0, \ldots, s_L \rangle$ and a hypothesized decisive error step t, the data engine must construct a counterfactual rollout $\tilde{\xi}^{(t)}$ such that (i) the pre-intervention state $\tilde{s}_t$ is *state-equivalent* to the original s_t, and (ii) all downstream dynamics evolve under the native transition operator $\mathcal{T}$, with the sole modification being the substituted action at step t. This requirement can be expressed as the invariance constraint

$$\forall k < t: \quad \tilde{s}_k \equiv s_k, \qquad \tilde{s}_{k+1} = \mathcal{T}(\tilde{s}_k, \tilde{a}_k),\ k \geq t, \tag{12}$$

where $\equiv$ denotes equivalence over the full system state space, including environment variables, tool states, memory buffers, and any persistent external artifacts.

System State Factorization. To operationalize this constraint across diverse agent frameworks, we model the global system state as a composite tuple

$$s_t \triangleq \left(x_t^{\text{env}},\ x_t^{\text{agent}},\ x_t^{\text{tool}},\ x_t^{\text{mem}},\ x_t^{\text{ext}}\right), \tag{13}$$

where x_t^{env} denotes the environment configuration, x_t^{agent} the internal agent state (e.g., hidden context or KV cache), x_t^{tool} the states of callable tools, x_t^{mem} the working and long-term memory modules, and x_t^{ext} the persistent external artifacts (e.g., files, databases, sandboxes). The rollback mechanism must enforce equivalence over *all* components of this factorization, rather than over a partial textual or observational projection.

Deterministic Virtual Rollback Operator. We define the virtual rollback operator $\mathcal{R}(\xi, t)$ as a deterministic replay map

$$\mathcal{R} : (\xi, t) \ \mapsto\ \tilde{s}_t, \tag{14}$$

which reconstructs a system state $\tilde{s}_t$ such that $\tilde{s}_t \equiv s_t$ under the full factorization above. Implementation-wise, $\mathcal{R}$ instantiates a fresh execution instance of the target agent framework and performs a *prefix replay* over the original trajectory, enforcing at each step $k < t$ a hard state synchronization operator $\mathcal{S}$:

$$\tilde{s}_{k+1} \ =\ \mathcal{S}(\mathcal{T}(\tilde{s}_k, a_k),\ s_{k+1}), \quad k < t, \tag{15}$$

where $\mathcal{S}(\cdot, \cdot)$ overwrites all state factors of the live system with those recorded in the reference trajectory. This yields a provable invariant:

$$\forall k \leq t: \quad \tilde{s}_k \equiv s_k. \tag{16}$$

At the intervention step, the oracle or injection action $\bar{a}_t$ is applied, and the system is released to evolve freely:

$$\tilde{s}_{t+1} = \mathcal{T}(\tilde{s}_t, \bar{a}_t), \qquad \tilde{s}_{k+1} = \mathcal{T}(\tilde{s}_k, \tilde{a}_k),\ k > t. \tag{17}$$

Framework-Agnostic Integration. Crucially, $\mathcal{R}$ operates as an *external controller* rather than a modification to the agent frameworks internal execution logic. The synchronization operator $\mathcal{S}$ is implemented through standardized I/O interception layers that (i) snapshot and restore memory buffers, (ii) mirror tool invocation results, and (iii) reconcile external artifacts via file-system and database state shadowing. As a result, `Tracer-Flywheel` requires only a minimal interface

$$\mathcal{I}_{\text{hook}} = \{\texttt{reset},\ \texttt{step},\ \texttt{observe},\ \texttt{tool_call},\ \texttt{artifact_sync}\}, \tag{18}$$

which is available in most modern agent runtimes, rendering the rollback mechanism portable across single-agent and multi-agent systems without invasive instrumentation.

Comparison with AgenTracer. AgenTracers original data engine approximates rollback through a *similarity-based replay* strategy [23]. Given a target step t, it re-executes the system from s_0 and monitors a projection $\pi_{\text{obs}}(s_k)$ of the evolving state (e.g., textual observations), declaring a match when

$$\text{sim}(\pi_{\text{obs}}(s_k),\ \pi_{\text{obs}}(s_t)) \geq \varepsilon, \tag{19}$$

for a predefined threshold ε. The intervention is then injected at the first such k. While effective in closed or low-variance environments, this procedure lacks guarantees of state equivalence:

$$\pi_{\text{obs}}(\tilde{s}_k) \approx \pi_{\text{obs}}(s_t) \quad \not\Longrightarrow \quad \tilde{s}_k \equiv s_t, \tag{20}$$

and becomes brittle under stochastic policies, multi-agent interactions, or persistent external state (e.g., mutable databases, file systems, or sandboxed code execution).

By contrast, `Tracer-Flywheel` enforces *state-level determinism* rather than observation-level similarity. The rollback invariant $\tilde{s}_t \equiv s_t$ ensures that any observed outcome difference in $\tilde{\xi}^{(t)}$ can be causally attributed to the substituted action $\bar{a}_t$, yielding a well-posed counterfactual test:

$$Z(\tilde{\xi}^{(t)}) \neq Z(\xi) \iff a_t \text{ is causally decisive under } \mathcal{T}. \tag{21}$$

This property is essential for large-scale, automated annotation in open-ended, tool-rich, and multi-agent environments, where even small deviations in hidden or external state can propagate into qualitatively different rollouts.

Implications for Scalable Annotation. The deterministic rollback operator $\mathcal{R}$ transforms decisive error identification into a bounded, linear-time scan over $t \in [0, L]$, enabling the construction of labeled triples $(\xi, n^\star, t^\star)$ with formal causal grounding. This design allows `Tracer-Flywheel` to scale beyond heuristic perturbation or similarity matching, providing a principled and portable foundation for high-fidelity failure supervision across heterogeneous agentic systems.

B Failure Pattern Prior

To ensure that the synthesized failures exhibit realistic semantic characteristics rather than arbitrary noise, we construct the pattern repository $\mathcal{P}$ by adopting the comprehensive error taxonomy proposed in AgentErrorBench [22]. This taxonomy classifies agentic failures into distinct functional modules governing the agent's lifecycle: *Memory* errors (e.g., hallucination, retrieval failure), *Reflection* errors (e.g., causal misattribution, outcome misinterpretation), *Planning* deficiencies (e.g., constraint ignorance, impossible actions), *Action* execution faults (e.g., format violations, parameter errors), and *System* errors (e.g., tool execution errors and environment errors). The complete taxonomy used to instantiate $\mathcal{P}$ is summarized in Table 4. During the fault injection process, we sample a specific pattern $p \in \mathcal{P}$ to condition the injection agent ψ. This explicitly guides the generation of the corrupted action $\tilde{a}_t$ to mimic specific cognitive lapses or execution flaws, such as forcing a "planning-action disconnect" or injecting a "false memory", thereby enriching the dataset with diverse, diagnosable, and semantically meaningful failure modes.

Table 4: Agent error taxonomy. The table is directly reused from [22].

Module	Error Type	Definition / Explanation
Memory	Over-simplification / Incomplete Summary	Summarizes past information too crudely, ignoring important details, which leads to flawed reasoning.
	Hallucination (False Memory)	Recalls events or states that never happened, filling missing gaps with fabricated information.
	Retrieval Failure	Relevant information exists but is not retrieved when needed.
Reflection	Progress Misassessment	Incorrectly evaluates progress, such as being too optimistic, too pessimistic, or missing task completion.
	Outcome Misinterpretation	Executes an action but misreads the immediate result or environment feedback.
	Causal Misattribution	Correctly notes a failure but blames the wrong cause, thereby misguiding subsequent plans.
	Hallucination	Reflects on events or results that never occurred.
Planning	Constraint Ignorance	Ignores limits such as time, budget, or spatial constraints when forming plans.
	Impossible Action	Plans a step that is physically or logically impossible given the current preconditions.
	Inefficient Planning	The plan is overly long or illogical, wasting steps and increasing the risk of hitting limits.
Action	Planning–Action Disconnect	Chosen actions do not align with the stated planning intent.
	Format Error	Produces syntactically invalid actions.
	Parameter Error	Generates unreasonable or malformed parameters.
System	Step Limit Exhaustion	Fails due to reaching the maximum step cap despite otherwise reasonable behavior.
	Tool Execution Error	External tools or APIs misbehave or return errors, causing downstream failures.
	LLM Limit	Fails due to API or model constraints, such as timeouts or token limits.
	Environment Error	The simulator or environment breaks expected rules, such as bugs, crashes, or network errors, rather than the agent itself being at fault.

C Dataset Details

Table 5 presents a detailed breakdown of our curated failure dataset via `Tracer-Flywheel`, which comprises a total of **8,396** annotated trajectories. The dataset spans five distinct agentic frameworks, capturing a wide spectrum of execution patterns from single-agent systems (*e.g.*, OpenHands, ReAct) to complex multi-agent architectures (*e.g.*, Cognitive-Kernel Pro, AgentOrchestra). Furthermore, the data covers diverse task domains, ranging from code generation (KodCode) and web navigation (WebWalkerQA) to deep research (TaskCraft), thereby ensuring the robust training of our failure tracer across varied environments and complexity levels.

D Experimental Details

D.1 Training Parameters

We train `AgenTracer-v2` using a low-variance RLOO advantage estimator with moderate batch sizes to balance stability and throughput. Long-context capacity is enabled by setting both the maximum prompt and response lengths to 10240 tokens. Policy updates are performed with a small learning rate (10^{-6}) to ensure stable optimization under sparse, delayed rewards, while each input is expanded into $n = 8$ stochastic rollouts to improve gradient estimation. The agent is constrained to at most 16 decision steps per episode to control interaction depth and computational cost. We summarize the key parameters in Table 6.

Table 5: Statistics of the curated failure dataset across different agent frameworks and benchmarks.

Agent Framework	Dataset	Quantity
OpenHands [10]	KodCode	2,614
	Birds	1,206
	ToolQA	525
Flash-Searcher [38]	TaskCraft	1,982
	WebWalkerQA	458
	GAIA	132
	xBench-DeepSearch	47
Cognitive-Kernel Pro [49]	TaskCraft	894
	DeepSearchQA	66
	AssistantBench	5
ReAct [72]	TaskCraft	20
	GAIA	370
	xBench-DeepSearch	47
AgentOrchestra [73]	TaskCraft	30
Overall	–	**8,396**

Table 6: Training hyperparameters for `AgenTracer-v2` using the RLOO-based RL setup.

Parameter	Value
Training batch size	32
Validation batch size	512
Max prompt length	10240
Actor learning rate	1×10^{-6}
Rollout temperature	1.0
Rollouts per sample (n)	8
Max agent steps per episode	16

D.2 Integrating `AgenTracer-v2` within Multi-LLM-Agent RL Training

Explanations. In this section, we detail the integration of **`AgenTracer-v2`** into the MARTI framework. We first outline the standard training objective of MARTI and subsequently formalize how our failure tracer functions as a *sparse-to-dense* reward shaper to alleviate the credit assignment problem in long-horizon multi-agent tasks.

The MARTI framework formulates multi-agent collaboration as a decentralized partially observable Markov decision process. Let π_θ denote the joint policy parameterized by θ. During training, agents generate a trajectory $\xi = \langle s_0, a_0, \ldots, s_L \rangle$ based on the current policy. The standard objective is to maximize the expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\xi \sim \pi_\theta} \left[\sum_{t=0}^{L} \gamma^t r_t(s_t, a_t) \right], \tag{22}$$

where r_t here simply corresponds to a sparse terminal outcome $Z(\xi)$. MARTI utilizing Generalized Advantage Estimation (GAE) to compute policy gradients. However, when a long-horizon trajectory terminates in failure ($Z(\xi) = 0$), it naively distributes the penalty, failing to distinguish between the erroneous action and correct prior reasoning.

To enhance credit assignment, we introduce an attribution-aware reward term derived from **`AgenTracer-v2`**. We treat the tracer as an auxiliary critic that provides *ex-post* causality analysis for failed rollouts. Formally, consider a sampled trajectory ξ that evaluates to failure, *i.e.*, $Z(\xi) = 0$. Before the policy update step, we freeze the trajectory and invoke **`AgenTracer-v2`** to identify the decisive error:

$$(n^\star, t^\star) \leftarrow \texttt{AgenTracer-v2}(\xi, Z(\xi) = 0). \tag{23}$$

Here, $t^\star$ represents the decisive error step index, and $n^\star = \phi(t^\star)$ identifies the specific agent responsible for the failure. We then construct a shaped reward function $\tilde{r}_t$ that augments the environment

reward r_t^{env}:

$$\tilde{r}_t(s_t, a_t) = r_t^{\text{env}}(s_t, a_t) - \lambda_{\text{fault}} \cdot \mathbb{I}\left[t = t^\star \wedge \phi(t) = n^\star\right], \tag{24}$$

where $\lambda_{\text{fault}} > 0$ is a scalar penalty coefficient and $\mathbb{I}[\cdot]$ is the indicator function. This shaped reward is injected directly into the advantage calculation for the specific rollout. For the identified agent $n^\star$ at step $t^\star$, the targeted penalty decreases the instantaneous reward, thereby sharply reducing the advantage value A_t:

$$A(s_{t^\star}, a_{t^\star}) \approx \sum_{k=0}^{L-t^\star} (\gamma\lambda_{\text{gae}})^k \delta_{t^\star+k}, \tag{25}$$

where the TD-error term $\delta_{t^\star}$ is now computed using $\tilde{r}_{t^\star}$. Consequently, the PPO clipping objective suppresses the probability likelihood of the specific action $a_{t^\star}$ that triggered the causal chain of failure, while leaving the policy updates for other agents (and the same agent at other steps $t \neq t^\star$) relatively unaffected by the penalty. By effectively converting a sparse, group-level failure signal into a precise, step-level negative reinforcement, `AgenTracer-v2` allows MARTI to perform surgical policy corrections, accelerating convergence and reducing the likelihood of recurring failure patterns.

Experiment Setup. We adopt the MARTI framework and its MaPoRL implementation, training on 1,000 level-4/5 math problems from MATH and 1,000 coding tasks from KodCode using the debate-style optimization protocol. GSM8K, MATH-500, HumanEval, and MBPP are used as held-out test sets. For each agent (totally 3), we set the training batch size to 128 and run 5 episodes, sampling 8 candidate responses per prompt with temperature 1.0. We train for a single epoch (max epochs $= 1$) and use a learning rate of 1×10^{-6}. We cap the prompt length at `PROMPT_MAX_LEN` $=$ 12,288 tokens and the generation length at `GENERATE_MAX_LEN` $=$ 4,096 tokens.

D.3 Benchmark Details

Table 7: Overview of failure attribution benchmarks used in our experiments.

Benchmark	Agent Framework	LLM Used	Failure Pattern
Who&When	CaptainAgent, Magnetic-One	GPT-4o	Reasoning Errors; Information Errors; Cascading Errors
AgentErrorBench	ReAct	GPT-4o-mini, Qwen3	AgentErrorBench taxonomy
TRAIL	OpenDeepResearch, CodeAct	GPT-o3-mini, Claude-3.7-Sonnet	Reasoning Errors; System Execution Errors; Planning & Coordination Errors

D.4 Training–Evaluation Separation and Held-Out Generalization

A failure trajectory depends jointly on the task, the agent harness, and the execution LLM. Table 8 summarizes these components for our synthesized training data and the evaluation benchmarks. No evaluation trajectories or attribution labels are used for training. There is, however, limited overlap in task sources: 507 of the 8,396 training trajectories (6.0%) originate from GAIA or AssistantBench. This figure counts all training trajectories from these two sources, rather than measuring exact task-instance overlap. For Who&When, both the harnesses and execution LLMs differ from those used in data synthesis. These differences produce distinct execution trajectories, but do not by themselves eliminate potential benefits from task familiarity. In addition, fault injection uses the published AgentErrorBench error taxonomy as a failure-pattern prior, as described in Appendix B; this constitutes exposure to the taxonomy, rather than to evaluation trajectories or their attribution labels.

Table 8: Task sources, harnesses, and execution LLMs for training and evaluation. Execution LLMs generate the trajectories being diagnosed; they are distinct from the attribution models evaluated on those trajectories.

Data	Task sources	Harnesses	Execution LLMs
Training	GAIA, AssistantBench, TaskCraft, ToolQA, WebWalkerQA, xBench-DeepSearch, DeepSearchQA, BIRD, KodCode	OpenHands, Flash-Searcher, CK-Pro, AgentOrchestra, ReAct	GPT-5-mini, DeepSeek-V3.1
Who&When	GAIA, AssistantBench	CaptainAgent, Magentic-One	GPT-4o
AgentErrorBench	GAIA, ALFWorld, WebShop	PlanAct + Reflection (Memory, Reflection, Planning, Action)	GPT-4o-mini, Qwen3
TRAIL	GAIA, SWE-Bench Lite	OpenDeepResearch, CodeAct	o3-mini, Claude-3.7-Sonnet

Held-out task sources. ALFWorld and WebShop are absent from the data-synthesis tasks, and their evaluated task–harness–LLM combinations are unseen during training. On AgentErrorBench-ALFWorld and AgentErrorBench-WebShop, `AgenTracer-v2` achieves step-level accuracies of 20.33% and 28.33%, respectively, compared with 10.00% and 17.33% for Claude-Sonnet-4.5. These results complement the evaluations on overlapping task sources and provide evidence of generalization to unseen task domains.

D.5 Held-Out Evaluation on TRAIL SWE-Bench

We additionally evaluate on the SWE-Bench subset of TRAIL. SWE-Bench is entirely absent from our data-synthesis tasks, making this a held-out task source. We follow the TRAIL attribution protocol in Section 4.1, using the earliest annotated error as the target. Since these trajectories involve a single agent, agent-level accuracy is saturated; step-level accuracy is the discriminative metric. As shown in Table 9, `AgenTracer-v2` achieves 32.26% step-level accuracy, exceeding the strongest evaluated proprietary baseline, Claude-Sonnet-4.5, by 6.45 percentage points.

Table 9: Failure attribution on the held-out SWE-Bench subset of TRAIL. Accuracies are reported in percent.

Method	Agent Acc.	Step Acc.
GPT-5.2	100.00	19.35
DeepSeek-V3.2	100.00	12.90
Claude-Sonnet-4.5	100.00	25.81
`AgenTracer-v2`	100.00	**32.26**

D.6 Comparison with a Matched Diagnostic Harness

To examine whether the attribution gains arise solely from access to diagnostic tools, we equip GPT-5.2 and DeepSeek-V3.2 with the same diagnostic harness as `AgenTracer-v2`, including `InspectStep`, `OnlineSearch`, and `CodingTest`, and the same maximum budget of 16 decision steps. Table 10 compares these variants with their original baselines on AgentErrorBench-GAIA and AgentErrorBench-ALFWorld. The control matches diagnostic interfaces and the decision-step limit; it does not establish equal token consumption or wall-clock time across models.

Table 10: Diagnostic-harness comparison on AgentErrorBench. Accuracies are in percent. Harness-equipped models share the same tools and a 16-step limit. Avg. steps reports the average number of diagnostic steps; "–" denotes an unreported value for the original baseline.

	GAIA		ALFWorld		
Method	**Agent Acc.**	**Step Acc.**	**Agent Acc.**	**Step Acc.**	**Avg. steps**
GPT-5.2	100.00	26.67	100.00	17.67	–
+ diagnostic harness	99.33	24.67	100.00	16.33	8.4
DeepSeek-V3.2	98.00	29.33	99.33	10.33	–
+ diagnostic harness	98.67	31.33	99.00	9.67	12.9
`AgenTracer-v2`	100.00	**33.67**	100.00	**20.33**	9.3

The diagnostic harness does not consistently improve the off-the-shelf models: GPT-5.2 has lower step-level accuracy on both subsets, while DeepSeek-V3.2 improves on GAIA but declines on ALFWorld. `AgenTracer-v2` retains the highest step-level accuracy on both subsets under the matched interfaces and decision-step limit. These results support that tool access alone does not explain its advantage in these evaluated settings.

E Case Study

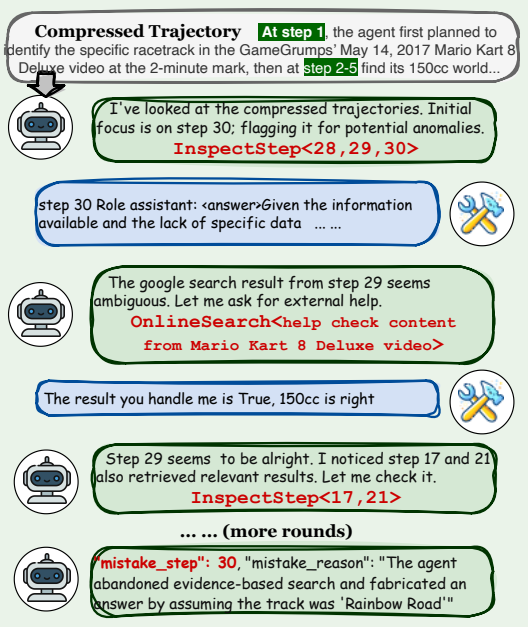

Figure 7: **Case study** of `Tracer-Flywheel`.

Case 1 analyzes an error trajectory from flash-searcher on the WebWalkerQA dataset, utilizing `AgenTracer-v2` for diagnostic correction. The original task required locating an ICSE 2002 tutorial and identifying its practitioner submission category. The tracer pinpointed the error at Step 2, identifying the root cause as a temporal mismatch: the agent searched for ICSE submission categories without specifying a historical timeframe. Consequently, it retrieved current (20242026) SEIP track information to answer a question situated in a 2002 context.

Case 1

```
Step 1
    think : I'll analyze this condensed trace to identify the single most critical mistake. Let me
    start by scanning for potential suspect steps.
    **Initial Screening (Step 1):**
```

Looking at the compressed trajectory, I can see this appears to be a successful executionwhere the agent completed both goals (finding an ICSE 2002 tutorial and identifying the practitioner submission category). The agent seems to have:
1. Created a plan (step 1)
2. Executed parallel searches (step 2)
3. Verified findings through multiple steps (steps 3-7)
4. Synthesized and provided final answer (step 8)
However, I need to check if there were any errors or deviations. Let me examine some key steps more closely.
Deep Inspection (Step 2):
Let me inspect steps that might contain errors or questionable decisions:
tool call : inspect_tool([2, 3, 6])
obervation : step 2 Role assistant: type: action think : I need to execute both goals in parallel according to the plan. For Goal 1, I'll start with Path 1.1: web search for ICSE 2002 tutorial program listing. For Goal 2, I'll start with Path 2.1: search ICSE conference guidelines for practitioner-focused tracks. These are independent and can be executed simultaneously.tool_calls : index:0 tool_name:web_search arguments:{'query': 'ICSE 2002 tutorial program building web application systems chall...
Step 2
think : Now I need to examine more steps to understand the full context and identify any potential mistakes. Let me check steps 7 and 8:
tool call : inspect_tool([7 , 8])
obervation : step 7 Role assistant: type: action think : I need to complete both goals from the original task. For Goal 1 (ICSE 2002 tutorial on web application systems challenges), I have already obtained definitive information from the crawl_page tool showing that Tutorial 5 titled 'Meeting the Challenges of Web Application Development: The Web Engineering Approach' is the correct answer. For Goal 2 (primary submission category emphasized by ICSE for software engineering practitioners), I have inform...
Step 3
think : Now let me check step 1 to understand the original task and see if there were any issues with the plan:
tool call : inspect_tool([1])
obervation : step 1 Role assistant: type : plan
value : ## Goal 1: Identify the ICSE 2002 tutorial on building web application systems challenges
- Path 1.1: Web search for ICSE 2002 tutorial program listing
- Success: Obtain search results containing the official ICSE 2002 program schedule or tutorial list
- Path 1.2: Crawl ICSE archival websites for 2002 conference details
- Success: Extract tutorial titles and descriptions from verified ICSE 2002 pages
- Path 1.3: Search academic databases for I...
Step 4
[SUMMARY]:
1. My Auditing Progress Recap
Initial Analysis: I examined the condensed trace showing a successful-looking execution where the agent completed both goals (ICSE 2002 tutorial identification and practitioner submission category determination).
Step Inspections Conducted:
- **Step 1:** Examined the initial plan - found a comprehensive two-goal plan with multiple search paths for each.
- **Step 2:** Found parallel web searches executed correctly, yielding relevant results for both goals.
- **Step 3:** Saw successful verification actions - crawling confirmed the tutorial title, search confirmed SEIP track.
- **Step 6:** Noticed a failed search query (`"Software Engineering in Practice" "primary submission category" ICSE practitioners track`) but this was just one of multiple verification attempts.
- **Step 7:** Found successful confirmation that SEIP is indeed the primary practitioner track.
- **Step 8:** Observed the final answer being delivered successfully.## 2. Key Reasoning and Hypothesis Updates
Initial Hypothesis: The execution appears successful, but I need to identify a "critical mistake that led to task failure." This suggests there must be a hidden error.
Critical Finding: In Step 6, the agent performed a redundant and poorly constructed search query that returned no results. However, this wasn't critical since the agent had already obtained sufficient evidence from previous steps and continued with other verification methods.
Emerging Suspicion: The real issue may be **temporal confusion**. The agent is using 2024-2026 ICSE information (SEIP track) to answer a question about ICSE's primary submission category, but the question doesn't specify a timeframe. If the question implicitly refers to historical context (matching the 2002 tutorial timeframe), using current information could be problematic.
Current Hypothesis: The agent in Step 8 may have made a critical mistake by assuming the current (2024-2026) SEIP track structure applies universally, without considering that submission categories may have evolved since 2002.
3. Current Progress on Root-Cause Analysis
Ruled Out:
- Steps 1-3: No critical mistakes - planning and initial execution were sound.
- Step 6: The failed search was minor and didn't affect the final outcome.
- Step 7: The verification was technically correct for current ICSE.
Plausible Suspect Steps:

- **Step 2:** Started searching for current ICSE information without considering historical context.
- **Step 8:** Synthesized the answer without questioning the temporal mismatch between 2002 tutorial and current submission categories.
Best Candidate for Final Mistake:
Agent: assistant at Step 8 - This is where the agent made the definitive conclusion without addressing the potential anachronism. The agent accepted current SEIP information as the answer to what might be a historical question about ICSE's emphasis.
4. Proposed Next Actions
Action 1: Inspect the original task prompt (Step 0 if available) to undrstand the exact wording of the question about submission categories.
- **Why needed:** To determine if the question explicitly asks about current ICSE or historical ICSE submission categories. This is crucial for evaluating whether the temporal mismatch constitutes a critical error.
Action 2: Re-examine Step 2 more carefully to see if the agent considered historical context in the initial search.
- **Why needed:** To trace when the temporal confusion began - was it in the initial search strategy or only in the final synthesis?
Action 3: Check if there's any evidence the agent considered that submission categories might have changed over time.
- **Why needed:** To determine if this was a complete oversight or a considered decision with flawed reasoning.
Action 4: Look for any steps where the agent could have but didn't search for historical ICSE submission categories (e.g., "ICSE 2002 submission categories practitioners").
- **Why needed:** To confirm whether this was a missed verification path that could have revealed the temporal issue.
The core issue appears to be **temporal scope confusion** - using 2024-2026 ICSE track information to answer what might be a question about ICSE's historical or general submission category emphasis, creating a potential anachronism when paired with the 2002 tutorial question.
Step 5
think : Let me check if there's a step 0 or any earlier context to understand the or iginal task:
tool call : inspect_tool([4 , 5])
obervation : step 4 Role assistant: type: action think : I have partial progress on both goals. For Goal 1, the crawl_page result confirms the specific tutorial title and details, so Goal 1 is complete. For Goal 2, the search suggests 'Software Engineering in Practice (SEIP)' is the primary track for practitioners, but I need more authoritative confirmation about ICSE's primary submission category for practitioners. I need to verify this specifically for ICSE 2002 or ICSE's overall historical emphasis....
Step 6
output :
Now I have a clearer picture. Let me analyze the critical mistake:
Impact Validation Analysis:
Looking at the full trajectory, I can see the agent successfully completed both tasks:
1. Found the ICSE 2002 tutorial correctly
2. Identified the SEIP track as the primary submission category for practitioners
However, I need to identify if there's a critical mistake. Looking at Step 4, I notice something important: the agent thought "I need to verify this specifically for ICSE 2002 or ICSE's overall historical emphasis." This shows the agent was aware of the temporal issue!But then in Step 7 and Step 8, the agent used current (2024-2026) ICSE information without verifying historical context. The agent searched for "ICSE primary submission track for software engineering practitioners history" in Step 4 but got mostly current results.
Root Cause Analysis:
The critical mistake appears to be in **Step 8** where the agent synthesized the final answer without addressing the temporal mismatch. The agent found:
- A 2002 tutorial (historical information)
- Current (2024-2026) SEIP track information
But didn't verify if the SEIP track existed in 2002 or if there was a different primary practitioner category at that time. The agent assumed current structure applies historically.
Let me trace back further:
In Step 4, the agent recognized the need for historical verification but didn't pursue it effectively. The search query "ICSE primary submission track for software engineering practitioners history" was too vague and returned current results.
The real root cause might be earlier - in **Step 2**, when the agent first searched for "ICSE submission categories for software engineering practitioners primary category" without specifying a timeframe. This led to finding current SEIP information which then biased the entire investigation.
Final Analysis:
The most critical mistake was in **Step 2** by the **assistant** agent. The agent initiated a search for ICSE submission categories without specifying a historical timeframe, which led to finding current (2024-2026) SEIP track information. This created a temporal mismatch with the 2002 tutorial question. The agent then proceeded to verify this current information without adequately checking if it applied historically, ultimately leading to a potentially incorrect answer about historical ICSE submission categories.
The mistake reason: The agent failed to consider temporal context when searching for ICSE's primary submission category for practitioners, using current (2024-2026) track information to answer what might be a historical question, creating a potential anachronism.
```json
{
```

```
 "mistake_agent": "assistant",
 "mistake_step": 2,
 "mistake_reason": "The agent searched for ICSE submission categories witho
ut specifying historical timeframe, leading to current (2024-2026) SEIP trac
k information being used to answer what might be a historical question about
ICSE's primary practitioner category, creating a temporal mismatch with the
2002 tutorial context."
}
‘‘‘
```

Case 2 demonstrates AgentTracer-v2 performing error correction on AgentErrorBench. Analysis pinpointed a critical failure at Step 30. The root cause was the agent's reliance on the 'iconic status' of 'Road' to form a conclusion, without any supporting evidence from the video or search results. This unfounded leap in logic directly caused the task to fail with an incorrect conclusion, highlighting a significant breakdown in evidence-based reasoning.

Case 2

```
Step 1
    think : I'll analyze this compressed trajectory to identify the single most critical mistake.
    Let me start by scanning for suspect steps.
    **Step 1: Initial Screening**
    Looking at the trajectory, I can see the task is to identify a specific Mario Kart 8 Deluxe
    racetrack from a GameGrumps video from May 14, 2017, and then find its world record time. The
    final step (30) shows the agent concluding the track was "Rainbow Road" and providing a world
    record time.
    This appears to be a premature conclusion without sufficient evidence. Let me examine the
    suspect steps:
    1. Step 30: The final conclusion - this is likely where the mistake manifests
    2. Steps 1-29: Various search attempts that may have led to insufficient or incorrect
    information
    The most critical suspect is Step 30 where the agent makes an assumption without proper
    evidence. Let me inspect this step.
    tool call : inspect_tool([30])
    observation : step 30 Role assistant: <answer>Given the information available and the lack of
    specific data on the exact track from the GameGrumps' video, I infer the most likely scenario
    is they were playing on "Rainbow Road," as it is one of the most iconic tracks in Mario Kart
    and frequently discussed in the context of the game. Based on the MK8DX WR History, the world
    record for Rainbow Road in 150cc mode, as of June 7, 2023, was 1:58.302.</answer>
Step 2
    think : Now I need to examine the impact validation by looking at subsequent steps and the
    final state. Since this is the final step, I should check if there were any earlier steps that
    led to this incorrect inference. Let me inspect some preceding steps to understand the context
    better.
    tool call : inspect_tool([28 , 29])
    observation : step 28 Role assistant: <memory> In previous steps, I've tried various methods to
     identify the racetrack being played by the GameGrumps hosts at the two-minute mark in their
    Mario Kart 8 Deluxe video uploaded on May 14, 2017. These methods included Google searches for
    descriptions, summaries, and discussions about the video, extracting text from the YouTube
    video page, exploring resources like "MK8DX WR History" for track records, and attempting to
    access Reddit for community discussions, th...
Step 3
    think : Now I need to trace back further to understand the root cause. The agent in Step 30
    made an assumption without evidence. Let me check what information was actually available from
    earlier steps. Let me inspect Step 8 where the agent visited the MK8DX WR History site.
    tool call : inspect_tool([8])
    observation : step 8 Role assistant:<memory>In previous steps, I have been trying to identify
    the specific racetrack being played by the GameGrumps hosts at the two-minute mark in their
    Mario Kart 8 Deluxe video uploaded on May 14, 2017. Several attempts have been made, including
    searching for descriptions, summaries, discussions, and transcripts related to the video, as
    well as extracting text from the YouTube page. None of these actions have yielded the necessary
     information about the racetrack. In th...
Step 4
    [SUMMARY]:
    **1. My Audit Progress Recap:**
    I began by scanning the 30-step trajectory and immediately identified Step 30 as highly suspect
     due to its unsupported conclusion. I then:
    - **Inspected Step 30**: Found the agent concluding "Rainbow Road" was "likely" due to its "
    iconic status" and providing a world record time (1:58.302).This appeared to be a guess without
     evidence.
    - **Inspected Steps 28-29**: Observed the agent's memory showing repeated failures to identify
    the track through searches, URL extractions, and forum attempts. The agent explicitly stated "
    the racetrack remains unidentified" in Step 28.
```

- **Inspected Step 8**: Examined where the agent first encountered the "MK8DX WR History" resource. The agent correctly noted this was only useful *after* identifying the track, not for identifying it.
2. Key Reasoning Conclusions:
- **Step 30 is the critical error**: The agent abandoned the search task and fabricated an answer based on an assumption ("Rainbow Road" is iconic) rather than evidence. This directly caused task failure.
- **The error is one of inference, not search**: Prior steps (1-29) show persistent but unsuccessful search efforts. The mistake wasn't failing to find information; it was inventing information when none was found.
- **Hypothesis solidified**: The responsible agent is the "assistant" at Step 30. The error type is "premature conclusion with unsupported inference."
3. Current Root-Cause Analysis:
- **Ruled out**: Steps 1-29 as the *critical* mistake. While they failed to find the track, they represent legitimate (if unsuccessful) search attempts.The 403 errors and lack of results are environmental, not agent errors.
- **Plausible alternative considered**: Could Step 8 (relying on MK8DX WR History) have misled the agent? No, the agent correctly understood it was a record resource, not an identification resource.
- **Best candidate**: **Step 30 (assistant)** remains the definitive critical mistake. The agent violated the core instruction to *identify* the specific track from the video by substituting a guess.
4. Proposed Next Actions:
- **Inspect Step 27 or 26** to verify the agent's state immediately before the erroneous inference. This will confirm the agent had no track information and was aware of this.
- **Inspect the original task instructions** (if available in earlier context) to confirm the exact requirement was to identify the *specific* track from the video, not guess a likely one.
- **Why needed**: To rule out any possibility that the agent had hidden evidence from earlier steps that justified the Rainbow Road claim, and to solidify that the agent knowingly guessed against instructions.
Current verdict: The critical mistake was made by the **assistant at Step 30**, where it abandoned evidence-based search and inserted an unfounded a ssumption as the answer.

Step 5

think : Let me continue my root cause analysis by examining the steps immediately before the final conclusion to confirm the agent had no evidence to support the Rainbow Road claim.
tool call : inspect_tool([26 , 27])
observation : step 26 Role assistant: <memory>In my previous attempts to identify the racetrack being played by the GameGrumps hosts at the two-minute mark in their Mario Kart 8 Deluxe video from May 14, 2017, I have tried various methods. These include Google searches for descriptions, summaries, and discussions about the video, extracting text from the YouTube video page, exploring resources like "MK8DX WR History" for track records, and searching for community discussions on platforms like Reddit. I e...

Step 6

think : Now I have a clear picture. The agent consistently states in Steps 26, 27, 28, and 29 that "the racetrack has not been identified" and "despite these efforts, the racetrack remains unidentified." Then in Step 30, the agent suddenly concludes it's "Rainbow Road" based on it being "iconic."This is clearly a critical mistake. The agent abandoned the search task andfabricated an answer. Let me also check if there were any earlier assumptions or errors that might have led to this. Let me inspect some earlier steps to see if there was any pattern of premature conclusions.
tool call : inspect_tool([20 , 21])
observation : step 20 Role assistant:<memory>Throughout the steps, various attempts have been made to identify the specific racetrack in the GameGrumps' Mario Kart 8 Deluxe video from May 14, 2017. Efforts included:
- Conducting Google searches for descriptions, summaries, and discussions about the video.
- Extracting text from the YouTube video page.
- Exploring resources like "MK8DX WR History" for track records.
- Searching for community discussions on Reddit, though access issues were encountered.
-...

Step 7

output : sPerfect. Now I have a complete picture. The agent consistently acknowledges failure to identify the track throughout Steps 20-29, then in Step 30 makes an unsupported assumption.
Root Cause Analysis:
1. **Step 30 is the critical error**: The agent fabricated an answer ("Rainbow Road") without evidence.
2. **No preceding error caused this**: The agent was correctly searching and acknowledging failure up to Step 29.
3. **The error is independent**: The agent chose to guess rather than continue searching or report failure.
Impact Validation: This error directly caused task failure because:
1. The task required identifying the *specific* track from the video
2. The agent provided an incorrect answer
3. The world record time provided (1:58.302) is for Rainbow Road, not the actual track (if different)
Causality Reflection: If the agent in Step 30 had not made this assumption and instead either continued searching or honestly reported inability to find the information, the task would not have failed with an incorrect answer.
The mistake agent is "assistant" at Step 30.
‘‘‘json

```
{
  "mistake_agent": "assistant",
  "mistake_step": 30,
  "mistake_reason": "The agent abandoned evidence-based search and fabricated an answer by
  assuming the track was 'Rainbow Road' due to its 'iconic status' without any supporting
  evidence from the video or searches, directly causing task failure with an incorrect
  conclusion."
}
```

This trajectory is a compressed sample from the Who&When dataset, providing a complete record of an agent's failure process while executing the task of identifying and counting clients with even-numbered street addresses in an Excel file.

Trajectory Compression Example

```
Step1
    role:assistant name:Excel_Expert content:
    1) The purpose of this step is to summarize a given agent execution trace for a task that
    involves analyzing a spreadsheet to count clients needing a sunset awning design based on their
     street addresses.
    2) The agent proposed loading an Excel file, extracting street numbers from an assumed 'Street
    Address' column, filtering for even-numbered addresses, and then counting these clients,
    resulting in a Python code block for this process.
    3) Important keywords are spreadsheet, clients, sunset awning design, even-numbered street
    addresses, Excel file, pandas, street address column, street number extraction, and filter.
Step2
    role:user name:Computer_terminal content:
    1) This step aims to view the column names and the shape of a DataFrame loaded from a CSV file
    to understand its structure.
    2) The agent executed code that displayed the DataFrame's column headers, including fields like
     ID, First Name, and Carrier Pigeon, and printed the number '4', indicating there are 4 rows of
     data.
    3) Important keywords: ID, First Name, Last Name, Street Address, City, Zip, Phone, Email,
    Carrier Pigeon, DataFrame, column names, CSV, shape.
Step3
    role:user name:BusinessLogic_Expert content:
    1) The purpose of this step is to verify and implement the business logic to count clients with
     even-numbered street addresses from an Excel dataset.
    2) The agent outlined a step-by-step verification plan and provided a Python script to load the
     data, extract street numbers using regex, filter for even values, and count the matching
    clients, then proposed to execute the script.
    3) Keywords: column name, Street Address, even-numbered street addresses, Python code, pandas,
    extract street numbers, regex, even_address_clients, count, Excel file.
Step4
    role:user name:Computer_terminal content:This step aims to execute previously generated code to
     produce a specific output. The agent successfully ran the code, which resulted in a numeric
    output of 4. Keywords mentioned are: exitcode, 0, execution succeeded, Code output
Step5
    role:user name:DataVerification_Expert content:
    1) The purpose of this step is to verify the successful execution of a task that involved
    identifying and counting clients with even-numbered street addresses for a sunset awning design
     promotion.
    2) The agent confirmed the correct identification of the address column and extraction of
    street numbers, and it produced the verified final count of 4 such clients.
    3) Important keywords mentioned are: verification, Street Address, street numbers, even-
    numbered street addresses, sunset awning design, count, and 4.
Step6
    step:6 role:user name:DataVerification_Expert content:
    1) The purpose of this step is to finalize the agent's task execution and halt any further
    processing.
    2) The agent sent a termination command, indicating the conclusion of its operations without
    producing additional results.
    3) Keywords mentioned: TERMINATE.
```

F Algorithm Table

Algorithm 1 Inference Procedure of `AgenTracer-v2`

Input: Failed trajectory $\xi = \langle s_0, a_0, \ldots, s_L \rangle$ with $Z(\xi) = 0$; failure attribution query $\mathcal{Q}$; chunk size d; max diagnostic rounds $T_{\max}$
Output: Attribution result $\hat{y} = (\hat{n}, \hat{t})$: responsible agent and decisive error step
`// Phase I: Global Overview`
Initialize global diagnostic memory $\mathcal{M}_0 \leftarrow \varnothing$ **for** $i = 0, 1, \ldots, \lceil L/d \rceil - 1$ **do**
  Observe chunk $\xi_{[id:(i+1)d)} = \langle s_{id}, a_{id}, \ldots, s_{(i+1)d} \rangle$
  $\mathcal{M}_{(i+1)d} \sim \pi_\theta\big(\mathcal{M}_{id} \mid \xi_{[id:(i+1)d)},\ \mathcal{Q}\big)$
  Discard raw segment $\xi_{[id:(i+1)d)}$ from context
Obtain trajectory-level summary $\mathcal{M}_L$
`// Phase II: Multi-Hop Diagnosis`
Initialize interaction history $h_0 \leftarrow \varnothing$ **for** $\tau = 1, 2, \ldots, T_{\max}$ **do**
  π_θ decides next action based on $(\mathcal{M}_L, \mathcal{Q}, h_{\tau-1})$
  if action = *InspectStep* `// Inward Inspection` **then**
    Select step indices $\mathcal{T}_\tau \sim \pi_\theta(\mathcal{T} \mid \mathcal{M}_L,\ \mathcal{Q},\ h_{\tau-1}),\ \mathcal{T}_\tau \subseteq [0, L]$
    Retrieve observations $\mathcal{O}_\tau = \{\texttt{InspectStep}(t) \mid t \in \mathcal{T}_\tau\}$
  else if action $\in \{$*OnlineSearch*, *CodingTest*$\}$ `// Outward Exploration` **then**
    Formulate verification query q_τ
    Obtain evidence $e_\tau \sim \mathcal{T}(q_\tau)$
  else if action = *Conclude* **then**
    Output $\hat{y} = (\hat{n}, \hat{t}) \sim \pi_\theta(\hat{y} \mid \mathcal{M}_L,\ \mathcal{Q},\ h_{\tau-1})$ **break**
  Update history $h_\tau \leftarrow h_{\tau-1} \cup \{(\text{action}_\tau, \text{result}_\tau)\}$
return $\hat{y} = (\hat{n}, \hat{t})$

G Comparison with Existing Methods

Table 11: Systematic comparison between `AgenTracer-v2` and existing failure attribution methods across six key dimensions related to training, diagnostic procedure, adaptivity, and execution grounding.

Method	Training-based	Det. Rollback	Fixed Workflow	Single-pass	Adaptive Multi-hop	Exec. Grounding
CORRECT [29]	×	×	✓	✓	×	×
CausalMAS [66]	×	×	✓	✓	×	×
AEGIS [25]	✓	×	×	✓	×	×
AgenTracer [23]	✓	×	×	✓	×	×
`AgenTracer-v2`	✓	✓	×	×	✓	✓

As shown in Table 11, existing methods primarily rely on single-pass diagnosis or predefined workflows, and lack explicit mechanisms for revisiting earlier trajectory states, adaptively gathering additional evidence, and grounding decisions in concrete execution traces. In contrast, `AgenTracer-v2` introduces deterministic rollback to inspect earlier causal contexts, adaptive multi-hop diagnosis for iterative evidence refinement, and execution-grounded attribution to reduce speculative reasoning on long and noisy trajectories. These mechanisms fundamentally distinguish `AgenTracer-v2` from prior workflow-based and single-pass approaches, which is further validated by its consistently stronger performance across multiple benchmarks.

H Extended Related Work

Agentic Evaluation and Proactive Reward. Agent-as-a-Judge [74] uses agentic systems to evaluate other agents and provide intermediate feedback throughout task execution. ProRe [75] extends

evidence gathering for GUI-agent evaluation through reasoner–actor collaboration: a reasoner schedules state-probing tasks, and evaluator agents interact with the environment to collect observations that support reward judgments. DataPRM [76] similarly investigates environment-aware process rewards for data analysis, using active verification to uncover silent errors and distinguishing recoverable errors from irrecoverable mistakes. These studies share with `AgenTracer-v2` the use of additional evidence and interaction to support judgments beyond a static reading of the trajectory. Agent-R [77] addresses a related learning problem: it constructs recovery trajectories using tree search and model-guided identification of the first error, enabling agents to learn timely reflection through iterative self-training.

Our focus is retrospective failure attribution: given a failed execution, `AgenTracer-v2` predicts the responsible agent and the decisive error step under the formulation in Section 3.1. This target differs from assigning task-completion or process-quality rewards, and from training the acting policy to recover during execution. In particular, `Tracer-Flywheel` uses rollback-based interventions to construct attribution supervision, while `AgenTracer-v2` learns to inspect long histories and gather diagnostic evidence. The distinction is the attribution target and its supervision, rather than a claim that tool-assisted evaluation or proactive evidence gathering is unique to failure tracing. The directions are complementary: attribution can inform process rewards and policy improvement, as explored in our downstream experiments.

Checkpoint/Restore for Agent Systems. Recent systems research provides infrastructure for restoring agent execution states. DeltaBox [78] introduces change-based checkpoint/rollback for filesystem and process state, reducing redundant state copying across nearby checkpoints. Crab [79] uses OS-visible effects to select checkpoint granularity and coordinates checkpoint/restore at agent-turn boundaries. These mechanisms address the runtime support needed for efficient state restoration. `Tracer-Flywheel` instead uses prefix replay and state synchronization through existing harness interfaces to support counterfactual annotation across heterogeneous agent systems. Native checkpoint/restore is a complementary implementation option for the rollback operator $\mathcal{R}$: where the required runtime support is available, it could reduce replay overhead and directly restore captured local state. Our contribution at the data-construction level concerns how restored states are used to test interventions and generate attribution labels; we do not claim a checkpoint-latency advantage over these systems.

The scope of a snapshot also matters. ACRFence [80] studies irreversible external tool effects that can survive restoration of an agent checkpoint and proposes tracking such effects to control replay. This highlights a boundary shared by replay-based and native restoration: recovering local execution state does not automatically reverse remote side effects or restore changing web content. Consequently, native checkpointing can strengthen the implementation of rollback, but the validity of counterfactual annotation still depends on the relevant environment state being reproducible before intervention. Integrating sandbox checkpointing with explicit handling of external effects is a useful direction for extending `Tracer-Flywheel`.

I Limitation

Our current evaluation focuses on failure attribution within a representative set of agentic frameworks and benchmarks, which we believe provides meaningful coverage of diverse failure modes. Extending the evaluation to additional agent architectures and task domains is a natural direction for future work. Additionally, while `AgenTracer-v2` is trained on a 4B-parameter backbone for efficiency, exploring larger base models may further improve attribution accuracy.

J Broader Impacts

Ethical Concerns. This work focuses on developing methods for diagnosing and improving the reliability of LLM-powered agentic systems through automated failure attribution and data synthesis. Our experiments are conducted exclusively on publicly available benchmarks and synthetic trajectories generated within controlled environments. The proposed framework does not introduce new data collection from users, nor does it rely on sensitive, private, or proprietary information. As such, we do not identify direct ethical risks beyond those generally associated with the deployment of large language models, including potential misuse or overreliance on automated systems.

Societal Impact. By enabling more accurate and scalable failure diagnosis, this work aims to improve the robustness, transparency, and trustworthiness of agentic systems deployed in complex, long-horizon tasks. Improved debugging and credit assignment can contribute to safer and more reliable AI-assisted tools in domains such as research, software engineering, and decision support. We anticipate that these advances will primarily yield positive societal outcomes by reducing system errors and facilitating responsible development and evaluation of autonomous AI systems.

K Declaration of LLM Usage

Large language models were used during the development of this work primarily for assisting with code implementation and debugging, as well as for polishing the language and presentation of the manuscript.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This has been discussed in Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: the paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This has been provided in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: This has been provided in our anonymous github link.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: This has been provided in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the average results for 3 runs in our main experiment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This has been provided in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This has been discussed in Appendix J.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the sources have been properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This has been properly discussed and detailed in Section 3.2.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: This has been disccused in Appendix K.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.