WHO&WHEN PRO: Can LLMs Really Attribute Failures in AI Agents?

Jiale Liu [1 2] **Huajun Xi** [3] **Shaokun Zhang** [1] **Yifan Zeng** [4] **Tianwei Yue** [5] **Chi Wang** [2] **Jian Kang** [3]
Qingyun Wu [1 2] **Huazheng Wang** [2 4]

Project page: https://whowhenpro.github.io

Abstract

Automated failure attribution (Zhang et al., 2025c) uses LLMs to identify where and why agentic systems fail. As agents become more capable, their failures become subtler, making automated attribution increasingly important. We introduce WHO&WHEN PRO, a large-scale benchmark for automated failure attribution in agentic systems. Using a strictly controlled pipeline that injects a failure only after exactly replaying a successful prefix, we construct 12,326 failed trajectories with golden labels across 3 modalities and 26 benchmarks covering various scenarios. Beyond benchmarking, we conduct extensive experiments and analyses, revealing systematic patterns in how models attribute failures across modalities, protocols, and model families, and providing empirical guidance for future automated failure attribution systems.

1. Introduction

Recent works show that model scaling (OpenAI, 2026; Anthropic, 2026; Google DeepMind, 2025), together with careful harness engineering (Lou et al., 2026; Lin et al., 2026; Pan et al., 2026; Lee et al., 2026), can produce AI agents capable of operating in diverse digital environments, with demonstrated potential across domains such as coding (Gao et al., 2025b; Yang et al., 2024; Xia et al., 2024), scientific discovery (Ren et al., 2025; Bran et al., 2023; Boiko et al., 2023), and complex real-world problem solving (Mialon et al., 2024; Liu et al., 2025b; Zhang et al., 2025a; Liu et al., 2026; Song et al., 2024).

However, advancing agentic systems becomes increasingly challenging as their capabilities grow: failures in more capable agents are more subtle and harder to detect, yet these failures provide crucial signals for further improvement. This motivates *automated failure attribution*, a research direction that uses LLMs to identify where and why agentic systems fail and convert failures into actionable feedback (Zhang et al., 2025c; Cemri et al., 2025). The goal is to make failure attribution benefit from model scaling in the same way agent capability does: *stronger models should support not only more capable agents, but also better detection of their increasingly subtle failures.* More broadly, such human-free feedback potentially provides active signals for building self-evolving agentic systems, thereby enabling agents to improve from their own failures (Gao et al., 2025a; Fang et al., 2025a).

As an early effort in this direction, Zhang et al. (2025c) introduced WHO&WHEN, a benchmark of 184 failure traces collected from 127 LLM agentic systems. Each trace identifies the failure agent, the error step, and the failure cause in natural language, defining a task of identifying **who** failed and **when**. However, WHO&WHEN narrows to text tasks, whereas real-world agentic deployments span much broader modalities, such as image/video reasoning. Its small scale of 184 instances is also insufficient to reflect the breadth and complexity of real-world agentic scenarios. At the core, both limitations arise from the difficulty of obtaining high-quality failure annotations without human intervention, a challenge that becomes even greater once moves outside the text domain.

To address these limitations, we introduce a scalable pipeline for automatically constructing failure attribution data. Specifically, we rollout agent trajectories on a broader range of multimodal agentic tasks and retain those that successfully complete the task. For each successful trajectory, we inject an error at a chosen step and then warm-start execution from that point to generate a failed continuation. Since the injected error is the only controlled change that turns a successful trajectory into a failed one, the resulting task failure can be attributed exactly to the injected agent and step, thus yielding golden labels under the decisive-error definition by (Zhang et al., 2025c).

Building on this pipeline, we introduce WHO&WHEN PRO, a large-scale benchmark for automated failure attribution in

[1]Penn State University [2]AG2ai, Inc. [3]Mohamed bin Zayed University of Artificial Intelligence [4]Oregon State University [5]Mathos AI. Correspondence to: Jiale Liu <jiale.liu@psu.edu>, Shaokun Zhang <svz5418@psu.edu>, Huazheng Wang <huazheng.wang@oregonstate.edu>.

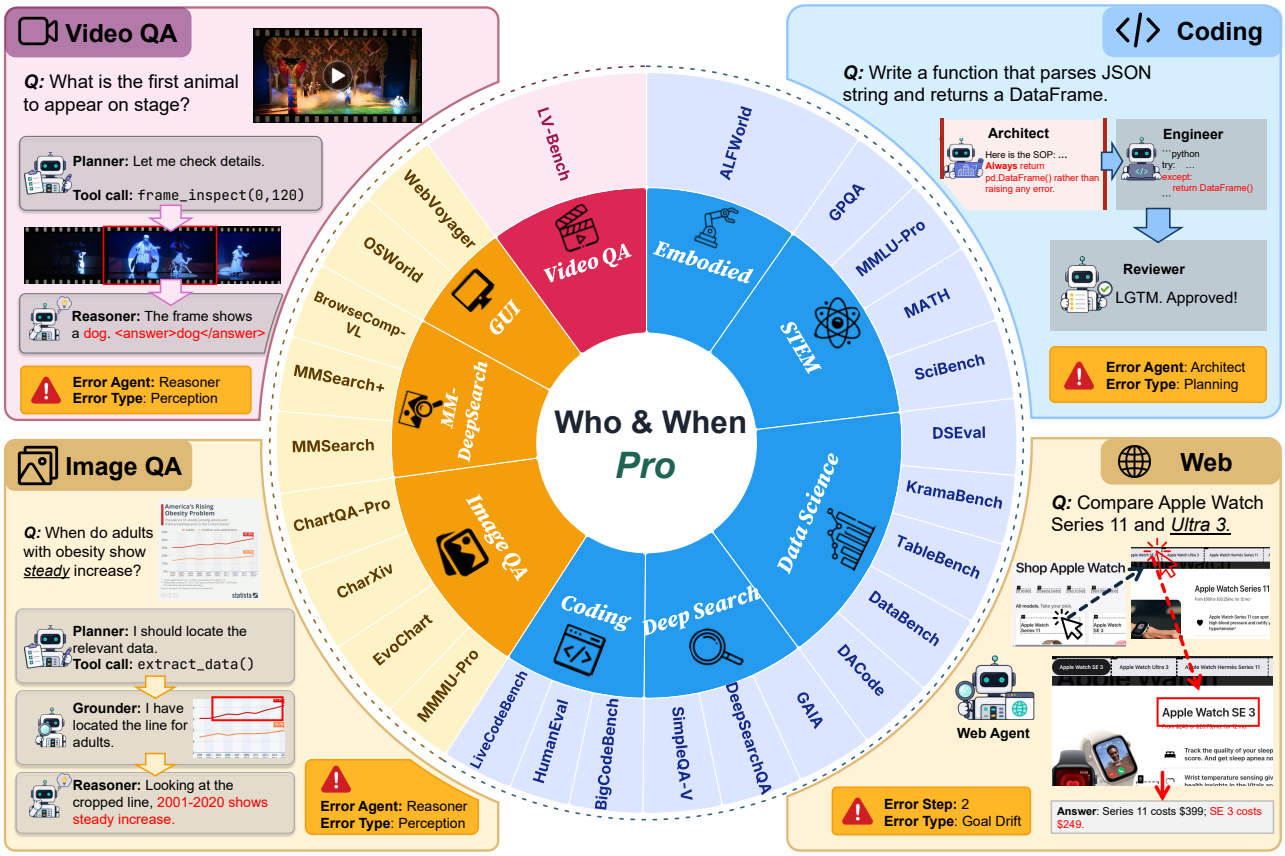


Figure 1. **Overview of WHO&WHEN PRO.** The inner ring groups 26 source benchmarks into 9 task categories. The outer ring maps each benchmark to its modality (**text**, **image**, **video**). Four corner panels illustrate failure attribution examples across modalities and agent topologies. **(Top left)** a **video** agent misidentifies a frame. **(Top right)** a multi-agent **coding** pipeline adopts a suboptimal architecture. **(Bottom left)** an **image** QA agent misreads a chart trend. **(Bottom right)** a **web** agent drifts from the user task. The decisive failure step are highlighted in each panel. Trace contents are summarized for illustration purposes and do not reflect the full agent outputs.

agentic systems. It contains 12,326 failure traces across 26 source benchmarks, 9 task categories, and 3 modalities, covering tasks from QA (Yue et al., 2025; Wang et al., 2024c) to data science (Wu et al., 2025; Hu et al., 2024; Lai et al., 2026), GUI interaction (Xie et al., 2024; He et al., 2024), and video understanding (Wang et al., 2024a). Compared with prior benchmarks, WHO&WHEN PRO substantially expands both scale and modality coverage, providing a more realistic and challenging testbed for evaluating failure attribution in modern agents.

Beyond benchmarking, we also perform extensive analyses showing that attribution behavior varies across modalities, protocols, and model families: video traces are especially challenging for step localization, multimodal traces provide stronger cues for failure-mode diagnosis, full-trajectory attribution outperforms incremental protocols, and open-weight models offer practical attribution backbones with favorable cost-performance trade-offs. Combined with previous works, WHO&WHEN PRO offers a rigorous foundation for measuring progress in automated failure attributions.

2. Related Work

2.1. Agents Failure Attribution

Failure attribution asks where, when, and by whom an agentic system deviates from the given task. It has rapidly emerged as a distinct field for understanding and optimizing agents. A first wave of benchmarks formalizes the joint prediction of the responsible agent and decisive step, scaling along complementary axes of expert curation (Zhang et al., 2025c; Cemri et al., 2025; Deshpande et al., 2025; Zhu et al., 2025; Barke et al., 2026) and LLM-driven injection (Kong et al., 2026; Zhang et al., 2026a). Another thread views attribution as inference over execution traces, pursued via hierarchical, spectrum-based (Ge et al., 2025b), representation learning based methods (Zhao et al., 2026), alongside lighter-weight runtime monitors and auditors (Zhu et al., 2026; Ma et al., 2026; Yu et al., 2025; Luo et al., 2026). Despite this breadth, every prior benchmark is text-only, isolates either the single-agent or multi-agent regime, and trades off label quality against scale. WHO&WHEN

Table 1. Comparison with existing failure attribution benchmarks. **Topology** indicates single-agent vs. multi-agent (MAS) traces. **Modality** indicates whether the benchmark covers text, image, or video agent traces.

	Scale			Topology		Modality		
Benchmark	**# Traces**	**# Benchmark**	**# Agents**	**Single**	**MAS**	**Text**	**Image**	**Video**
Who&When (Zhang et al., 2025c)	184	2	2		✓	✓		
MAST (Cemri et al., 2025)	1,642	9	7		✓	✓		
TRAIL (Deshpande et al., 2025)	148	2	2	✓		✓		
AgentErrorBench (Zhu et al., 2025)	500	3	3	✓		✓		
AEGIS (Kong et al., 2026)	9,533	6	6		✓	✓		
AgenTracer (Zhang et al., 2026a)	2,476	7	6		✓	✓		
AgentRx (Barke et al., 2026)	115	3	3	✓	✓	✓		
WHO&WHEN PRO (Ours)	**12,326**	**26**	**15**	✓	✓	✓	✓	✓

PRO bridges these gaps: it spans text, image, and video across 26 benchmarks, covers both regimes, and yields labels with higher-fidelity compared to other automatic failure generation pipeline. We provide a comparison between WHO&WHEN PRO and previous works in Table 1.

2.2. Self-Evolving Agents

Self-evolving agents hold the promise to learn from their own trajectories without fine-tuning, as charted by recent surveys (Gao et al., 2025a; Fang et al., 2025b; Liu et al., 2025a). Classical building blocks for self-refinement combine verbal reflection (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024), self-rewarding judgement (Yuan et al., 2024), refinement tuning (Fu et al., 2025; Yuan et al., 2025; Zeng et al., 2023), and bootstrapped multi-agent improvement (Zhao et al., 2025). Recent works close the loop more aggressively with attribution-aware (Zhai et al., 2025; Yang et al., 2026), trajectory-level (Lin et al., 2025; Chen et al., 2025; Ge et al., 2025a; Acikgoz et al., 2025; Xia et al., 2025), or skill-level self-evolution (Zhang et al., 2026b; 2024). A common bottleneck is that the feedback signal is either coarse or self-generated, both of which propagate the very errors the agent should be learning to avoid. WHO&WHEN PRO provides the missing intermediate signal: large-scale, externally verified, step-level failure labels across modalities, directly suited to drive self-evolution loops rather than terminal-only or self-graded ones.

3. WHO&WHEN PRO

We adopt the decisive-error formulation of Zhang et al. (2025c). Let $\tau = (a_1, \ldots, a_T)$ denote an agent trajectory, where a_t is the action produced by one agent at step t. The **decisive step** t^* is the earliest index such that correcting a_{t^*} would turn a failed trajectory into a successful one. In multi-agent traces, the **decisive agent** is the agent that produced a_{t^*}. We construct decisive-error labels at scale through controlled error injection on warm-started rollouts. In the remainder of this section, we cover the construction of source trajectories (§ 3.1), the failure-mode taxonomy (§ 3.2), the error-injection pipeline (§ 3.3), benchmark statistics (§3.4), and human validation (§3.5).

3.1. Source Trajectories

To support failure attribution across the diversity of modern agent deployments, we source trajectories from 15 agentic frameworks and 26 benchmarks spanning text, image, and video modalities. The frameworks range from single-agent ReAct loops (Yao et al., 2022; Roucher et al., 2025) to multi-agent pipelines with specialized roles (Hong et al., 2023; Wu et al., 2024; Zhang et al., 2025b), covering both code-as-action and tool-calling paradigms. The benchmarks span 9 task categories, from STEM QA to data science, GUI interaction, and video understanding, ensuring that the resulting failure traces reflect the breadth of scenarios where agents are deployed in practice.

For each (agent, benchmark) pair, we run the agent and log the complete execution trace, including task inputs, agent messages, observations, intermediate artifacts.[1] We evaluate each final answer using the benchmark's official evaluator when available. The resulting trajectories are partitioned into successful and failed runs.

The two partitions serve complementary roles. Successful trajectories are used as *seed traces*: they provide action steps that are known to be sufficient for task completion. Failed trajectories are not directly used as labeled benchmark items. Instead, they are used to calibrate the failure taxonomy and

[1]For a full list of agentic systems, benchmarks used, please refer to Appendix G, H.

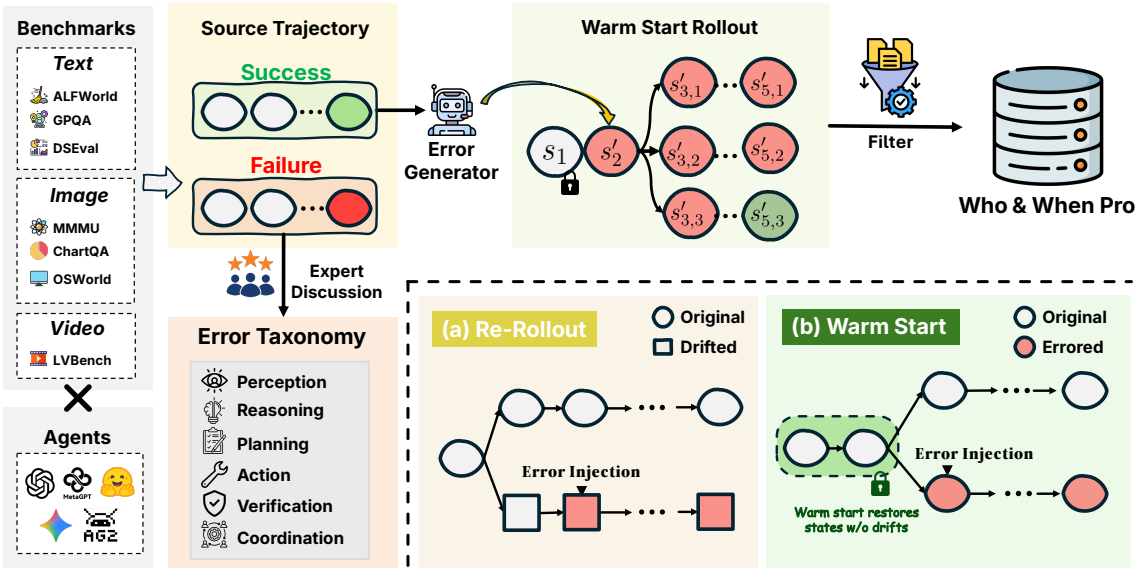


Figure 2. **WHO&WHEN PRO failure-injection pipeline.** Agent trajectories collected from benchmarks across text, image, and video modalities are split into successes and failures, with failures used to build an error taxonomy. For each successful seed, an error generator injects a taxonomy-conditioned mistake and a warm-start rollout collects the traces that turned to failure. **(Bottom right)** (a) Previous methods using re-rollout can cause pre-injection drift, leading to the derived decisive error label approximate; (b) We avoid this problem with warm start-based rollout.

to determine which error modes are realistic for each (agent, benchmark) setting, as described next.

3.2. Failure Taxonomy

We build the failure taxonomy from naturally failed traces. For each (agent, benchmark) pair, Ph.D.-level reviewers inspect sampled failures step by step, identify the earliest decisive error following the definition of Zhang et al. (2025c). Reviewers then compare recurring patterns across benchmarks and consolidate them into a unified taxonomy spanning perception, reasoning, planning, action, verification, and coordination failures. The full taxonomy and definitions are provided in Appendix C.

The review stage also produces a *failure-mode profile* for each (agent, benchmark) pair. A profile records which failure naturally appear in that setting and where they can plausibly arise. These profiles constrain the failure generation in Sec 3.3: human reviewers define the space of plausible failures and their applicable settings, while automated generation introduces those failures in seed traces. This separation keeps the benchmark scalable without allowing unconstrained synthetic failures to define the label space.

3.3. Scalable Failure Trace Generation

A core challenge in synthetic failure generation is label fidelity. Prior injection-based methods re-run the task from scratch and inject error during the new rollout (Kong et al., 2026; Zhang et al., 2026a). However, LLM generation is non-deterministic, new rollouts can drift from the original before the injection point (Figure 2, bottom right), making the label approximate. We instead warm-start from a successful seed $\tau = (a_1, \ldots, a_T)$. The pipeline selects an injection step t, constructs an erroneous action $\tilde{a}_t$, *replays* $(a_1, \ldots, a_{t-1})$ to restore the agent's context and the environment state, substitutes $\tilde{a}_t$ for a_t, and lets the original system run from step $t{+}1$ onward. If the resulting trajectory fails, it enters the benchmark: reverting $\tilde{a}_t$ recovers the successful seed, so t is the decisive step by definition.

Choosing the injection step For each originally correct trace, we sample t conditioned on the target failure mode. The sampler excludes invalid steps, such as steps with execution errors or steps where the target mode cannot naturally occur, and uses position preferences for different error families. For example, planning errors are usually injected earlier, while verification errors are injected after partial evidence has been collected, perception errors prefer steps where agent receives image observation.

Constructing the error step We construct $\tilde{a}_t$ in two stages to keep the error on-policy. A frontier model reads the seed's context up to step t, the task, and the target failure mode, and generates an adaptive injection prompt. This prompt is then patched into the agent's model call at step t, and the base agent model generates $\tilde{a}_t$ from its original context augmented with the injection prompt, preserving stylistic consistency with the agent's own behavior.

Table 2. **The overall statistics of WHO&WHEN PRO.**

Statistic	Value
# Traces	12,326
# Source Benchmarks	26
# Task Categories	9
# Agent Frameworks	15
# Error Modes	18
Avg. / Max trace length	7.5 / 50 steps
Avg. words / trace	1,139
Avg. words / step	152
Avg. images / trace[2]	1.3

Warm-start and post-injection rollout We warm-start the agent by replaying $(a_1, \ldots, a_{t-1})$ through the agent framework. Each replayed action passes through the framework's pipeline, including tool parsing, code execution, and environment interaction, progressively rebuilding context and the environment state to match the successful seed at step t. We distinguish two classes of environment interaction for replay fidelity. *Static tool calls* (web search, page retrieval) are served from a content-addressable cache populated during seed collection, guaranteeing byte-identical observations. *Stateful environments* that cannot be cached, such as live browser sessions, are instead re-executed with action-level fidelity checks that abort the attempt if the observed state diverges from the seed (details in Appendix F). At step t, the framework processes $\tilde{a}_t$ through the same pipeline. From step $t+1$ onward the system resumes with the same agent settings. All post-injection behavior is produced by the agent reacting to the consequences of a single injected action.

Post-hoc Filtering We retain only trajectories where the post-injection rollout ends in task failure. The surviving traces pass through additional quality filters. We remove traces where the content of $\tilde{a}_t$ leaks construction artifacts, such as references to the injection prompt. We also filter traces where the correct answer of the task is already salient in the agent's context before step t, which would make the injected error implausible to an external observer.

3.4. Benchmark Statistics

Applying this pipeline across all available (agent, benchmark) pairs yields WHO&WHEN PRO. It contains 12,326 failure traces spanning 26 benchmarks, 9 task categories, 3 modalities, 15 agent frameworks. Table 2 and Figure 3 summarize the benchmark composition. For a more detailed breakdown, we defer readers to Appendix D.5.

[2]Calculated on multimodal traces only.

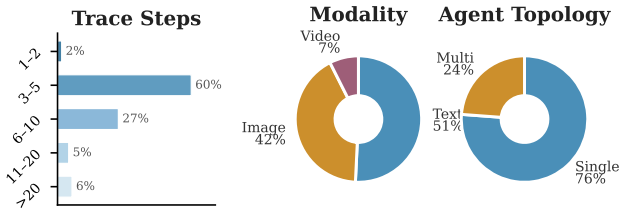


Figure 3. **Dataset composition.** Distributions of trace length, modality, and agent topology.

Human approved (%)			No clear	Agreement κ
Step	Agent	Error	(%)	
94.0	90.0	90.0	2.0	0.73

Table 3. **Results of human approval rate on 100 stratified traces.** Agent-level approval rate is calculated on MAS traces only. κ: Fleiss' kappa on error family.

3.5. Human Review

Three independent annotators review 100 traces stratified across modalities and error families. Each annotator sees the task, reference answer, full trajectory, and generated label, and either accepts it or provides a corrected step and error family, or marks it as having no clear decisive error.

Table 3 reports majority-vote acceptance rates. Annotators confirm 94% of step labels and 90% of both agent and error-family labels, with inter-annotator agreement of Fleiss' κ=0.73 (substantial). Only 2% of traces are flagged by a majority as having no clear decisive error, indicating that the injected failures are recognizable to human reviewers in nearly all cases. We further provide transfer matrix in Figure 11 in Appendix for analysis.

4. Experiments

4.1. Experiment Setup

Methods Following Zhang et al. (2025c), we evaluate three attribution methods. In **All-at-Once**, the LLM receives the full trajectory in one prompt and predicts the responsible agent, error step, and failure mode jointly. In **Step-by-step**, the LLM reads the trajectory sequentially and stops once it believes the first decisive error has occurred. In **Binary-search**, the LLM recursively compares trajectory segments to localize the decisive step. Unless otherwise stated, we use All-at-once and do not provide the task's ground-truth answer during attribution.

Metrics We report four complementary metrics. **Agent** is the accuracy of identifying the responsible agent and is computed only on multi-agent traces. **Step** is the accuracy of localizing the first decisive error step. It is scored as exact

Table 4. **Failure attribution performance of different LLMs on WHO&WHEN PRO.** We adopt the all-at-once protocol for evaluation. Best per column in **bold**, second-best underlined. "—" denotes the modality is not supported by the LLM and we cannot benchmark it on the multi-modal traces.

Model	Text				Image				Video			
	Agent	Step	Error	Joint	Agent	Step	Error	Joint	Agent	Step	Error	Joint
Closed-source models												
GPT-5.4	55.7	72.3	15.3	21.3	67.7	**65.2**	26.5	**30.2**	**85.1**	60.5	33.8	21.6
Claude Sonnet 4.6	54.9	69.8	19.1	22.4	67.5	63.8	**29.5**	29.6	79.8	**64.4**	37.4	26.4
Gemini 3 Flash	52.1	71.2	16.2	17.3	62.5	63.9	26.0	29.6	74.4	59.8	**40.0**	23.0
Grok 4.1	51.1	65.9	18.6	19.9	67.7	60.5	21.4	26.4	70.1	53.4	36.7	23.8
Open-weight models												
gpt-oss-120b	48.4	63.9	10.8	16.2	—	—	—	—	—	—	—	—
GLM-5	54.9	71.1	**22.2**	**25.3**	—	—	—	—	—	—	—	—
DeepSeek-V4-pro	52.0	68.7	16.6	17.5	—	—	—	—	—	—	—	—
Gemma 4	52.8	68.4	16.1	17.1	69.0	62.6	19.1	24.3	71.5	58.7	35.7	27.7
Llama-4 Maverick	51.9	66.7	20.6	16.5	61.0	54.9	20.6	19.0	79.8	59.5	36.4	**30.4**
Qwen3.5-122B	**57.5**	**73.9**	17.0	21.6	**70.4**	60.9	21.8	24.5	74.9	60.3	35.2	16.1

match against the dataset label. **Error** is macro-F1 over failure-mode classes. **Joint** is the fraction of traces where Agent, Step, and Error are all correct simultaneously.

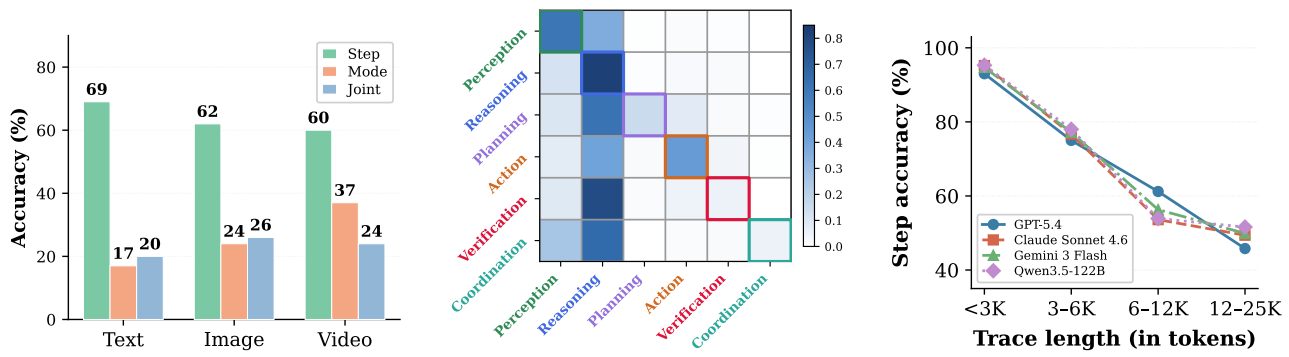
Models We evaluate both close-source and open-weight frontier LLMs on WHO&WHEN PRO. Closed-source models include **GPT-5.4** (OpenAI, 2026), **Claude Sonnet 4.6** (Anthropic, 2026), **Gemini 3 Flash** (Google DeepMind, 2025), and **Grok 4.1** (xAI, 2025). Open-weight models include **gpt-oss-120b** (OpenAI, 2025c), **GLM-5** (GLM-5 Team, 2026), **DeepSeek-V4-Pro** (DeepSeek-AI, 2026), **Gemma 4** (Google DeepMind, 2026), **Llama-4 Maverick** (Meta, 2025), and **Qwen3.5-122B** (Qwen Team, 2026). All traces fit in the native context window of every evaluated model, we present the traces in full without truncation or summarization.

4.2. Main Results

Accurate failure attribution remains a challenge. Table 4 reports results for ten models across all four metrics. The best model reaches 73.9% step-level attribution accuracy on text but only 22.2% error-mode F1, and similar gaps persist across image and video. No single model dominates across metrics or modalities. Qwen3.5-122B leads on agent localization, GPT-5.4 is strongest on step identification, and GLM-5 achieves the highest Joint accuracy on text. This pattern suggests that failure attribution exercises a fundamentally different capability from general knowledge answering or instruction following, one that current frontier models have not yet saturated.

Step localization and mode classification are bottlenecked by different modalities. Figure 4a shows that step accuracy is highest on text (69%) but drops on image and video, where longer multimodal context makes localization harder. Mode classification shows the opposite trend, rising from 17% on text to 37% on video, as visual and behavioral cues provide diagnostic signal that pure text traces lack. A text-only benchmark would mask this difficulty profile entirely.

Models classify errors by surface-level similarity rather than tracing root causes. The confusion matrix (Figure 4b) reveals that perception, reasoning, and action errors are reliably recognized because they leave clear signatures in the observations or tool calls. Planning, verification, and coordination errors, however, are frequently mislabeled as reasoning errors. These failures originate early in the trace but propagate through later steps, producing symptoms that resemble reasoning mistakes by the time they become visible. For example, an ineffective plan leads to a sequence of individually plausible but ultimately misguided steps, and a skipped verification allows an early mistake to compound uncorrected. In both cases, the local evidence at the point of failure looks like bad reasoning, even though the root cause lies further back. This pattern suggests that current models classify based on the most salient symptom rather than tracing the causal chain back to the decisive step, where more than half of other errors on multimodal traces are mislabeled



(a) Attribution accuracy by modality (b) Failure mode confusion matrix (c) Step accuracy vs. trace length

Figure 4. **Analysis of attribution difficulty. (a)** Step accuracy drops sharply on video traces while mode classification improves on multimodal traces. **(b)** Failure mode confusion matrix on multimodal traces, averaged over seven vision-capable models. **(c)** Step accuracy for four representative models, grouped by trace length in tokens.

as reasoning.

Longer traces are harder to diagnose. We then grouped the traces according to length and plot the step-level attribution accuracy. As shown in Figure 4c, step accuracy falls from 94% on short traces under 3K tokens to 50% on traces exceeding 12K tokens. For illustration purpose, we select the representative four models. However, this drop is consistent across all ten models, pointing to a general difficulty with long contexts rather than a weakness of any particular model. The sharpest decline is between 3–6K and 6–12K tokens, roughly where traces transition from single tool calls to multi-step sequences with interleaved observations. As more correct steps accumulate around the error, isolating the decisive error becomes harder.

4.3. Comparison between Attribution Methods

For the subsequent ablation studies, which require running different LLMs under multiple protocol and variants, we use a stratified subset of 1,444 traces that preserves the distribution over modalities and error categories. This keeps the analysis faithful on a representative slice of the data.

Full-trajectory attribution consistently outperforms incremental methods. Table 6 compares all-at-once, step-by-step, and binary search on this subset. All-at-once achieves the highest Joint accuracy for eight of ten models and improves Step accuracy over step-by-step by 7.6 percentage points, confirming that seeing the complete trajectory before making a decision benefits both localization and diagnosis.

Step-by-step attribution suffers from early commitment. In Figure 5, step-by-step has the lowest mean prediction distance on the failed attribution traces, confirming that models systematically flags a step *before* the actual error. This is because the evaluator processes steps sequentially, it locks onto the first suspicious step and cannot revise after observing later context. Binary search is directionally unbiased but equally imprecise, as slicing can remove the relevant context needed for accurate localization.

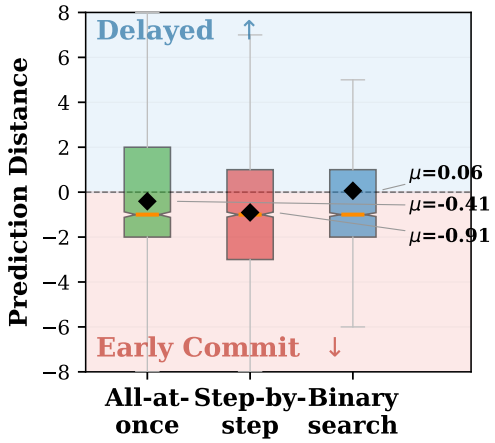


Figure 5. **Distribution of prediction distance using different methods.** Distances are calculated by subtracting the LLM-predicted error step from the ground-truth step. Diamonds (⧫) mark the mean.

4.4. Cost-Performance Trade-off

We investigate the cost of failure attribution under each protocol with different models[3] in Figure 6. All-at-once dominates the Pareto frontier across all three modalities. On text, the frontier consists entirely of all-at-once configurations, and on image only GPT-5.4 under binary search exceeds it. Notably, open-weight models match or surpass

[3]For closed-source models, we calculate costs using their official providers' pricing. For open-weight models, costs are calculated by OpenRouter's API per-token pricing.

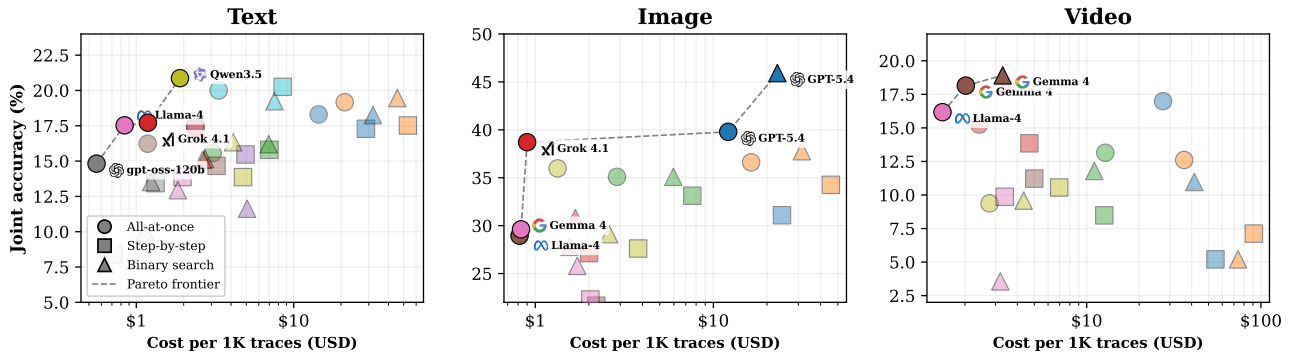
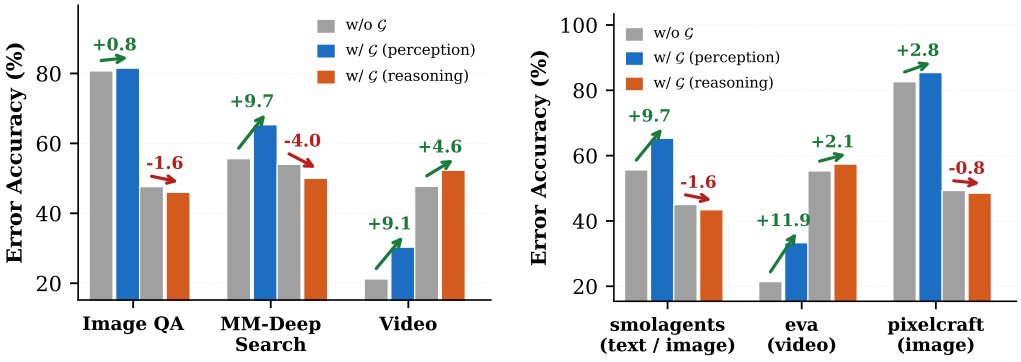


Figure 6. **Cost vs. joint accuracy across attribution protocols.** Each point is one (model, protocol) pair on the ablation subset. Shapes denote protocols; colors denote models. The dashed line marks the Pareto frontier. All-at-once dominates the low-cost regime in all three modalities.



(a) Impact of $\mathcal{G}$ by task domain

(b) Impact of $\mathcal{G}$ by agentic framework

Figure 7. **Impact of ground truth on error mode classification.** Per-trace accuracy for perception (blue) and reasoning (orange) errors, evaluated on the multimodal part of the ablation subset. Arrows indicate the change from w/o $\mathcal{G}$ to w/ $\mathcal{G}$.

closed-source alternatives at a fraction of the cost. Qwen3.5 achieves the highest text accuracy at seven times cheaper than GPT-5.4, and Gemma 4 leads on video. Failure attribution thus admits a favorable cost-accuracy trade-off, with open-weight models offering practical alternatives to frontier closed-source systems.

4.5. Impact of Ground Truth

We test whether providing the task's ground-truth answer (w/ $\mathcal{G}$) helps diagnose errors, mirroring verifier-assisted self-improvement where outcome feedback is available. Figure 7 reveals a split along error type: perception-error attribution improves while reasoning-error attribution degrades or stays flat, across all task domains and frameworks.

This reflects an asymmetry between *outcome* and *process* verification. Perception errors like wrong OCR and misread charts are hard to verify without a reference (Zhang et al., 2025d), so $\mathcal{G}$ fills a genuine gap. Reasoning errors leave a traceable logical chain. Providing $\mathcal{G}$ tempts the judge to shortcut via answer comparison rather than tracing process, which is less reliable when different failure modes arrive at similar final answers. Appendix B illustrates both failure patterns with paired judge rationales.

5. Conclusion

We introduce WHO&WHEN PRO, a large-scale benchmark for failure attribution in LLM agents, comprising 12,326 traces across 26 source benchmarks spanning text, image, and video modalities and covering both single- and multi-agent settings. The benchmark is built via a warm-started error-injection pipeline that produces golden labels for the responsible agent, decisive step, and failure mode. Evaluating frontier closed- and open-source LLMs reveals that current models still struggle with attribution, particularly in diagnosing failure modes and localizing errors in long multimodal traces, with attribution behavior varying systematically across modalities, protocols, and model families. We hope WHO&WHEN PRO will serve as a rigorous foundation for future work on reliable agent debugging and self-improving agentic systems.

References

Acikgoz, E. C., Qian, C., Ji, H., Hakkani-Tür, D., and Tur, G. Self-improving llm agents at test-time. *arXiv preprint arXiv:2510.07841*, 2025.

Anthropic. System Card: Claude Sonnet 4.6, February 2026. URL https://anthropic.com/claude-sonnet-4-6-system-card. Official system card. Accessed: 2026-05-05.

Awadallah, A., Lara, Y., Magazine, R., Mozannar, H., Nambi, A., Pandya, Y., Rajeswaran, A., Rosset, C., Taymanov, A., Vineet, V., Whitehead, S., and Zhao, A. Fara-7b: An efficient agentic model for computer use. *arXiv:2511.19663*, 2025.

Banerjee, A., Nair, A., and Borogovac, T. Where did it all go wrong? a hierarchical look into multi-agent error attribution. *arXiv preprint arXiv:2510.04886*, 2025.

Barke, S., Goyal, A., Khare, A., Singh, A., Nath, S., and Bansal, C. Agentrx: Diagnosing ai agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, December 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06792-0. URL https://doi.org/10.1038/s41586-023-06792-0.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., and Stoica, I. Why do multi-agent LLM systems fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=fAjbYBmonr.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, Y., Wang, Y., Zhu, S., Yu, H., Feng, T., Zhang, M., Patwary, M., and You, J. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. Accessed: 2026-05-05.

Deshpande, D., Gangal, V., Mehta, H., Krishnan, J., Kannappan, A., and Qian, R. Trail: Trace reasoning and agentic issue localization, 2025. URL https://arxiv.org/abs/2505.08638.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

Fang, J., Peng, Y., Zhang, X., Wang, Y., Yi, X., Zhang, G., Xu, Y., Wu, B., Liu, S., Li, Z., et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025a.

Fang, J., Peng, Y., Zhang, X., Wang, Y., Yi, X., Zhang, G., Xu, Y., Wu, B., Liu, S., Li, Z., et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025b.

Fourney, A., Bansal, G., Mozannar, H., Tan, C., Salinas, E., Niedtner, F., Proebsting, G., Bassman, G., Gerrits, J., Alber, J., et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Fu, D., He, K., Wang, Y., Hong, W., GongQue, Z., Zeng, W., Wang, W., Wang, J., Cai, X., and Xu, W. Agentrefine: Enhancing agent generalization through refinement tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=FDimWzmcWn.

Gao, H.-a., Geng, J., Hua, W., Hu, M., Juan, X., Liu, H., Liu, S., Qiu, J., Qi, X., Wu, Y., et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1, 2025a.

Gao, P., Tian, Z., Meng, X., Wang, X., Hu, R., Xiao, Y., Liu, Y., Zhang, Z., Chen, J., Gao, C., et al. Trae agent: An llm-based agent for software engineering with test-time scaling. *arXiv preprint arXiv:2507.23370*, 2025b.

Ge, Y., Romeo, S., Cai, J., Sunkara, M., and Zhang, Y. SAMULE: Self-learning agents enhanced by multi-level reflection. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16591–16610, Suzhou, China, November 2025a. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.839. URL https://aclanthology.org/2025.emnlp-main.839/.

Ge, Y., Xie, L., Li, Z., Pei, Y., and Zhang, T. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. *arXiv preprint arXiv:2509.13782*, 2025b.

Geng, X., Xia, P., Zhang, Z., Wang, X., Wang, Q., Ding, R., Wang, C., Wu, J., Li, K., Zhao, Y., Yin, H., Jiang, Y., Xie, P., Huang, F., Yao, H., Fung, Y. R., and Zhou, J. Webwatcher: Breaking new frontiers of vision-language deep research agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8jsaazdAb3.

GLM-5 Team. GLM-5: from Vibe Coding to Agentic Engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3 Flash Model Card, December 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf. Official model card. Accessed: 2026-05-05.

Google DeepMind. Gemma 4 Model Card, April 2026. URL https://ai.google.dev/gemma/docs/core/model_card_4.

Gou, Z., Shao, Z., Gong, Y., yelong shen, Yang, Y., Duan, N., and Chen, W. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Grijalba, J. O., Ureña-López, L. A., Cámara, E. M., and Camacho-Collados, J. Question answering over tabular data with databench: A large-scale empirical evaluation of llms. In *Proceedings of LREC-COLING 2024*, Turin, Italy, 2024.

Guo, X., Huang, K., Liu, J., Fan, W., Vélez, N., Wu, Q., Wang, H., Griffiths, T. L., and Wang, M. Embodied llm agents learn to cooperate in organized teams. *arXiv preprint arXiv:2403.12482*, 2024.

Gupta, N., Chatterjee, R., Haas, L., Tao, C., Wang, A., Liu, C., Oiwa, H., Gribovskaya, E., Ackermann, J., Blitzer, J., et al. Deepsearchqa: Bridging the comprehensiveness gap for deep research agents. *arXiv preprint arXiv:2601.20975*, 2026.

Haas, L., Yona, G., D'Antonio, G., Goldshtein, S., and Das, D. Simpleqa verified: A reliable factuality benchmark to measure parametric knowledge, 2025. URL https://arxiv.org/abs/2509.07968.

He, H., Yao, W., Ma, K., Yu, W., Dai, Y., Zhang, H., Lan, Z., and Yu, D. WebVoyager: Building an end-to-end web agent with large multimodal models. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Hu, X., Zhao, Z., Wei, S., Chai, Z., Wang, G., Wang, X., Su, J., Xu, J., Zhu, M., Cheng, Y., Yuan, J., Kuang, K., Yang, Y., Yang, H., and Wu, F. Infiagent-dabench: Evaluating agents on data analysis tasks, 2024.

Huang, J.-t., Zhou, J., Jin, T., Zhou, X., Chen, Z., Wang, W., Yuan, Y., Lyu, M. R., and Sap, M. On the resilience of llm-based multi-agent collaboration with faulty agents. *arXiv preprint arXiv:2408.00989*, 2024a.

Huang, M., Han, L., Zhang, X., Wu, W., Ma, J., Zhang, L., and Liu, J. Evochart: A benchmark and a self-training approach towards real-world chart understanding, 2024b. URL https://arxiv.org/abs/2409.01577.

Huang, Y., Luo, J., Yu, Y., Zhang, Y., Lei, F., Wei, Y., He, S., Huang, L., Liu, X., Zhao, J., and Liu, K. Dacode: Agent data science code generation benchmark for large language models, 2024c. URL https://arxiv.org/abs/2410.07331.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Jiang, D., Zhang, R., Guo, Z., Wu, Y., Lei, J., Qiu, P., Lu, P., Chen, Z., Song, G., Gao, P., et al. Mmsearch: Benchmarking the potential of large models as multimodal search engines. *arXiv preprint arXiv:2409.12959*, 2024.

Kong, F., Zhang, R., Yin, H., Zhang, G., Zhang, X., Chen, Z., Zhang, Z., Zhang, X., Zhu, S.-C., and Feng, X. Aegis:

Automated error generation and attribution for multi-agent systems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zqcYoxXiN3.

Lai, E., Vitagliano, G., Zhang, Z., Chabra, O., SUDHIR, S., Zeng, A., Zabreyko, A. A., Li, C., Kossmann, F., Ding, J., Chen, J., Markakis, M., Russo, M., Wang, W., Wu, Z., Cafarella, M., Cao, L., Madden, S., and Kraska, T. KRAMABENCH: A benchmark for AI systems on data-to-insight pipelines over data lakes. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fZfUdeCC5X.

Lee, Y., Nair, R., Zhang, Q., Lee, K., Khattab, O., and Finn, C. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Li, Y., Inan, H. A., Yue, X., Chen, W.-N., Wutschitz, L., Kulkarni, J., Poovendran, R., Sim, R., and Rajmohan, S. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025.

Lin, J., Guo, Y., Han, Y., Hu, S., Ni, Z., Wang, L., Chen, M., Jiang, D., Jiao, B., Hu, C., and Wang, H. Se-agent: Self-evolution trajectory optimization in multi-step reasoning with llm-based agents. *arXiv preprint arXiv:2508.02085*, 2025. URL https://arxiv.org/abs/2508.02085.

Lin, J., Liu, S., Pan, C., Lin, L., Dou, S., Huang, X., Yan, H., Han, Z., and Gui, T. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

Liu, B., Li, X., Zhang, J., Wang, J., He, T., Hong, S., Liu, H., Zhang, S., Song, K., Zhu, K., et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025a.

Liu, J., Zeng, Y., Zhang, S., Zhang, C., Højmark-Bertelsen, M., Gadeberg, M. N., Wang, H., and Wu, Q. Divide, optimize, merge: Scalable fine-grained generative optimization for LLM agents. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 18990–19012, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1034. URL https://aclanthology.org/2025.findings-emnlp.1034/.

Liu, J., Bursztyn, V., Ai, L., Wang, H., Choudhary, S., Mitra, S., and Wu, Q. Teamfusion: Supporting open-ended teamwork with multi-agent systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14435–14456, 2026.

Liu, Z., Zhang, Y., Li, P., Liu, Y., and Yang, D. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=XII0Wp1XA9.

Lou, X., Lázaro-Gredilla, M., Dedieu, A., Wendelken, C., Lehrach, W., and Murphy, K. P. Autoharness: improving llm agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026.

Luo, H., Dai, S., Ni, C., Li, X., Zhang, G., Wang, K., Liu, T., and Salam, H. Agentauditor: Human-level safety and security evaluation for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=2KKqp7MWJM.

Ma, M., Zhang, J., Yang, F., Kang, Y., Lin, Q., Rajmohan, S., and Zhang, D. Dover: Intervention-driven auto debugging for LLM multi-agent systems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mrEK16Jy6h.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Majgaonkar, O., Fei, Z., Li, X., Sarro, F., and Ye, H. Understanding code agent behaviour: An empirical study of success and failure trajectories. *arXiv preprint arXiv:2511.00197*, 2025.

Masry, A., Islam, M. S., Ahmed, M., Bajaj, A., Kabir, F., Kartha, A., Laskar, M. T. R., Rahman, M., Rahman, S., Shahmohammadi, M., et al. Chartqapro: A more diverse and challenging benchmark for chart question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19123–19151, 2025.

Meta. Llama 4 Model Card, April 2025. URL https://github.com/meta-llama/llama-models/blob/main/models/llama4/MODEL_CARD.md.

Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., and Scialom, T. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

OpenAI. Computer-Using Agent. https://openai.com/index/computer-using-agent/, January 2025a. Accessed: 2026-05-06.

OpenAI. GPT-4.1, April 2025b. URL https://openai.com/index/gpt-4-1/. Accessed: 2026-05-07.

OpenAI. gpt-oss-120b & gpt-oss-20b Model Card, 2025c. URL https://arxiv.org/abs/2508.10925.

OpenAI. GPT-5.4 Thinking System Card, March 2026. URL https://openai.com/index/gpt-5-4-thinking-system-card/. Official system card. Accessed: 2026-05-05.

Pan, L., Zou, L., Guo, S., Ni, J., and Zheng, H.-T. Natural-language agent harnesses. *arXiv preprint arXiv:2603.25723*, 2026.

Qian, C., Xie, Z., Wang, Y., Liu, W., Zhu, K., Xia, H., Dang, Y., Du, Z., Chen, W., Yang, C., Liu, Z., and Sun, M. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=K3n5jPkrU6.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5. Official model-specific card for Qwen3.5-122B-A10B: https://huggingface.co/Qwen/Qwen3.5-122B-A10B. Accessed: 2026-05-05.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Ren, S., Xie, C., Jian, P., Ren, Z., Leng, C., and Zhang, J. Towards scientific intelligence: A survey of llm-based scientific agents. *arXiv preprint arXiv:2503.24047*, 2025.

Roucher, A., del Moral, A. V., Wolf, T., von Werra, L., and Kaunismäki, E. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Shridhar, M., Yuan, X., Cote, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Song, L., Liu, J., Zhang, J., Zhang, S., Luo, A., Wang, S., Wu, Q., and Wang, C. Adaptive in-conversation team building for language model agents. *arXiv preprint arXiv:2405.19425*, 2024.

Song, L., Dai, Y., Prabhu, V., Zhang, J., Shi, T., Li, L., Li, J., Savarese, S., Chen, Z., Zhao, J., Xu, R., and Xiong, C. Coact-1: Computer-using agents with coding as actions, 2025. URL https://arxiv.org/abs/2508.03923.

Song, Y., Ramaneti, K., Sheikh, Z., Chen, Z., Gou, B., Xie, T., Xu, Y., Zhang, D., Gandhi, A., Yang, F., Liu, J., Ou, T., Yuan, Z., Xu, F. F., Zhou, S., Wang, X., Yue, X., Yu, T., Sun, H., Su, Y., and Neubig, G. Agent data protocol: Unifying datasets for diverse, effective fine-tuning of LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tG6301ORHd.

Tao, X., Teng, Y., Su, X., Fu, X., Wu, J., Tao, C., Liu, Z., Bai, H., Liu, R., and Kong, L. Mmsearch-plus: A simple yet challenging benchmark for multimodal browsing agents. *arXiv preprint arXiv:2508.21475*, 2025.

Wang, W., He, Z., Hong, W., Cheng, Y., Zhang, X., Qi, J., Huang, S., Xu, B., Dong, Y., Ding, M., and Tang, J. Lvbench: An extreme long video understanding benchmark, 2024a.

Wang, X., Hu, Z., Lu, P., Zhu, Y., Zhang, J., Subramaniam, S., Loomba, A. R., Zhang, S., Sun, Y., and Wang, W. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=bq1JEgioLr.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024c. URL https://openreview.net/forum?id=y10DM6R2r3.

Wang, Z., Xia, M., He, L., Chen, H., Liu, Y., Zhu, R., Liang, K., Wu, X., Liu, H., Malladi, S., Chevalier, A., Arora, S., and Chen, D. Charxiv: Charting gaps in realistic chart

understanding in multimodal LLMs. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024d. URL https://openreview.net/forum?id=cy8mq7QYae.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Wu, X., Yang, J., Chai, L., Zhang, G., Liu, J., Du, X., Liang, D., Shu, D., Cheng, X., Sun, T., et al. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25497–25506, 2025.

Wu, Y., Jia, F., Zhang, S., Li, H., Zhu, E., Wang, Y., Lee, Y. T., Peng, R., Wu, Q., and Wang, C. Mathchat: Converse to tackle challenging math problems with llm agents. *arXiv preprint arXiv:2306.01337*, 2023.

xAI. Grok 4.1 Model Card, November 2025. URL https://data.x.ai/2025-11-17-grok-4-1-model-card.pdf. Official model card. Accessed: 2026-05-05.

Xia, C. S., Deng, Y., Dunn, S., and Zhang, L. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

Xia, C. S., Wang, Z., Yang, Y., Wei, Y., and Zhang, L. Live-swe-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint*, 2025.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

Xu, Y., Lu, D., Shen, Z., Wang, J., Wang, Z., Mao, Y., Xiong, C., and Yu, T. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EEgYUccwsV.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Yang, K., Liu, Y., Chaudhary, S., Fakoor, R., Chaudhari, P., Karypis, G., and Rangwala, H. Agentoccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=oWdzUpOlkX.

Yang, R., Chen, H., Zhang, J., Zhao, M., Qian, C., Wang, K., Wang, Q., Koripella, T. V., Movahedi, M., Li, M., et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025b.

Yang, S., Ma, Z., Huang, T., Hu, Y., Wang, Y., and Chu, X. Coevolve: Training llm agents via agent-data mutual evolution. *https://arxiv.org/abs/2604.15840*, 2026.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yu, Y., Li, M., Xu, S., Fu, J., Hou, X., Lai, F., and Wang, B. Correct: Condensed error recognition via knowledge transfer in multi-agent systems. *arXiv preprint arXiv:2509.24088*, 2025.

Yuan, S., Chen, Z., Xi, Z., Ye, J., Du, Z., and Chen, J. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

Yuan, W., Pang, R. Y., Cho, K., Li, X., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Yue, X., Zheng, T., Ni, Y., Wang, Y., Zhang, K., Tong, S., Sun, Y., Yu, B., Zhang, G., Sun, H., Su, Y., Chen, W., and Neubig, G. MMMU-pro: A more robust multi-discipline multimodal understanding benchmark. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15134–15186, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.736. URL https://aclanthology.org/2025.acl-long.736/.

Zeng, A., Liu, M., Lu, R., Wang, B., Liu, X., Dong, Y., and Tang, J. Agenttuning: Enabling generalized agent abilities for llms, 2023.

Zhai, Y., Tao, S., Chen, C., Zou, A., Chen, Z., Fu, Q., Mai, S., Yu, L., Deng, J., Cao, Z., Liu, Z., Ding, B., and Zhou, J. Agentevolver: Towards efficient self-evolving agent system, 2025. URL https://arxiv.org/abs/2511.10395.

Zhang, G., Wang, J., Chen, J., Zhou, W., Wang, K., and YAN, S. Agentracer: Who is inducing failure in the LLM agentic systems? In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=l05DseqvuD.

Zhang, H., Fan, S., Zou, H. P., Chen, Y., Wang, Z., Zhou, J., Li, C., Huang, W.-C., Yao, Y., Zheng, K., et al. Evoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026b.

Zhang, S., Zhang, J., Liu, J., Song, L., Wang, C., Krishna, R., and Wu, Q. Offline training of language model agents with functions as learnable weights. In *Forty-first International Conference on Machine Learning*, 2024.

Zhang, S., Dong, Y., Zhang, J., Kautz, J., Catanzaro, B., Tao, A., Wu, Q., Yu, Z., and Liu, G. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025a.

Zhang, S., Li, Z., Zhang, Y., Fu, J., Song, L., Bian, J., Zhang, J., Yang, Y., and Wang, R. Pixelcraft: A multi-agent system for high-fidelity visual reasoning on structured images. *arXiv preprint arXiv:2509.25185*, 2025b.

Zhang, S., Yin, M., Zhang, J., Liu, J., Han, Z., Zhang, J., Li, B., Wang, C., Wang, H., Chen, Y., and Wu, Q. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=GazlTYxZss.

Zhang, X., Zhang, X., Wu, Y., Cao, Y., Zhang, R., Chu, R., Yang, L., and Yang, Y. Generative universal verifier as multimodal meta-reasoner. *arXiv preprint arXiv:2510.13804*, 2025d.

Zhang, X., Jia, Z., Guo, Z., Li, J., Li, B., Li, H., and Lu, Y. Deep video discovery: Agentic search with tool use for long-form video understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026c. URL https://openreview.net/forum?id=oQYq9L1NVT.

Zhang, Y., Wang, R., Wang, J., Tang, Y., Zheng, X., Duan, H., Lu, H., Deng, H., and Lu, L. Eva: Efficient reinforcement learning for end-to-end video agent, 2026d. URL https://arxiv.org/abs/2603.22918.

Zhang, Z., Wang, Z., Zhang, X., Guo, Z., Li, J., Li, B., and Lu, Y. Infiniteweb: Scalable web environment synthesis for gui agent training, 2026e. URL https://arxiv.org/abs/2601.04126.

Zhao, W., Yuksekgonul, M., Wu, S., and Zou, J. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2025.

Zhao, X., Liu, S., Zhang, Y., Ma, Q., Cheng, G., Wang, N., and Liu, C. Promas: Proactive error forecasting for multi-agent systems using markov transition dynamics. *arXiv preprint arXiv:2603.20260*, 2026.

Zhou, P., Ai, J., Zhao, W., Wang, K., Peng, X., Shao, W., Yao, H., Zhang, K., et al. Mpbench: A comprehensive multimodal reasoning benchmark for process errors identification. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21586–21606, 2025.

Zhou, Y., Zhang, L., Wu, Y., Liu, J., Fan, X., Zhao, Z., and Yan, H. Synthetic sandbox for training machine learning engineering agents. *arXiv preprint arXiv:2604.04872*, 2026.

Zhu, C., Hong, S., Wu, J., Chawla, K., Tang, Y., Yin, Y., Wolfe, N., Babinsky, E., and Liu, D. Raffles: Reasoning-based attribution of faults for llm systems. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7659–7688, 2026.

Zhu, K., Liu, Z., Li, B., Tian, M., Yang, Y., Zhang, J., Han, P., Xie, Q., Cui, F., Zhang, W., et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

Zhuo, T. Y., Chien, V. M., Chim, J., Hu, H., Yu, W., Widyasari, R., Yusuf, I. N. B., Zhan, H., He, J., Paul, I., Brunner, S., GONG, C., Hoang, J., Zebaze, A. R., Hong, X., Li, W.-D., Kaddour, J., Xu, M., Zhang, Z., Yadav, P., Jain, N., Gu, A., Cheng, Z., Liu, J., Liu, Q., Wang, Z., Lo, D., Hui, B., Muennighoff, N., Fried, D., Du, X., de Vries, H., and Werra, L. V. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

Zhuo, T. Y., Wang, D., Ding, H., Kumar, V., and Wang, Z. Cyber-zero: Training cybersecurity agents without runtime. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1gRTeAik4G.

A. Limitations and Broader Impacts

A.1. Limitations

Taxonomy scope. The 18-mode taxonomy was developed through iterative analysis of agent traces across 26 benchmarks. While it covers the dominant failure patterns we observed across text, image, video, single-agent, and multi-agent settings, we do not claim exhaustive coverage of all possible agent failure modes. As agent capabilities expand into new domains (e.g., more embodied and collaborative settings (Yang et al., 2025b; Guo et al., 2024)), additional failure categories may become necessary. The taxonomy is designed to be extensible, and the evaluation framework supports adding new modes without modifying existing annotations.

A.2. Broader Impacts

WHO&WHEN PRO is a diagnostic benchmark designed to improve the transparency and reliability of LLM-based agents. By providing fine-grained failure annotations, it enables researchers to identify systematic weaknesses in agent systems before deployment, supporting safer and more predictable agent behavior in practice. The error taxonomy and evaluation protocol are intended to benefit the broader community by establishing a shared vocabulary for agent failures and a reproducible methodology for measuring attribution quality. We release all traces, labels, and evaluation code to facilitate open research.

We acknowledge that the error injection methodology, while used here for constructive benchmarking, could in principle inform adversarial attacks against deployed agent systems. However, the failure modes we study (reasoning errors, planning failures, coordination breakdowns) reflect well-documented weaknesses that are already observable in production settings. Our contribution is not to surface new attack vectors but to provide structured ground truth that enables systematic measurement and, ultimately, mitigation of these failures. We believe the benefit of enabling rigorous failure attribution research substantially outweighs the marginal risk, particularly as agents are increasingly deployed in high-stakes applications where undiagnosed failures carry real consequences.

The benchmark does not involve human subjects, and all source trajectories are derived from publicly available academic benchmarks. No personally identifiable information is present in the dataset.

B. Case Studies

We present qualitative case studies that illustrate how providing ground truth $\mathcal{G}$ changes the failure-attribution model's diagnostic reasoning. Each case shows a complete agent trace alongside paired judge rationales under the w/o $\mathcal{G}$ and w/ $\mathcal{G}$ conditions.

B.1. Case 1: Ground Truth Shifts Diagnosis from Process to Outcome Verification

Figure 8 presents a trace from the MMSearch benchmark in which a smolagents-based system is asked when Pokémon Trading Card Game Pocket will arrive. The agent successfully retrieves the correct release date (October 30, 2024) in Step 2 from the official Pokémon website. However, in the injected Step 3, the agent reinterprets the user's question: instead of reporting the launch date, it treats "arrive" as referring to upcoming anniversary content planned for late October 2025. The ground-truth error mode is therefore *task misunderstanding*, because the agent answered a fundamentally different question than what was asked, despite having the correct information available.

The diagnosis comparison in Figure 8 reveals the mechanism by which $\mathcal{G}$ degrades error classification. Without $\mathcal{G}$, both GPT-5.4 and Claude Sonnet 4.6 perform process-level verification: they trace the agent's reasoning chain and recognize that the agent "structurally misunderstood" the question by redefining what "arrive" means. With $\mathcal{G}$ (the gold answer 2024-10-30), both judges shift to outcome-level verification: they notice that the agent "had already retrieved the correct release date" matching $\mathcal{G}$ and conclude that the agent "misapplied that evidence," predicting *reasoning error* instead. The agent did not misapply a correct fact; it answered an entirely different question. The gold answer created a value-matching shortcut that obscured this semantic failure.

B.2. Case 2: Ground Truth Causes the Judge to Skip the Root-Cause Error

Figure 9 presents a 4-step trace where the agent is asked when the glasses shown in an image were released. The image clearly displays "GENTLE MONSTER" branding, and the agent correctly identifies this brand in Step 1. However, in the

Query: "When will Pokémon Trading Card Game Pocket arrive?"
Gold answer: 2024-10-30
GT error mode: *Task Misunderstanding* (the agent misinterprets the user's intent and answers a different question)

Step	Agent Action and Observation
1	`web_search`("Pokémon Trading Card Game Pocket release date") *Obs:* 1. Pokémon TCG Pocket Celebrates Its First Anniversary… Releasing on Oct. 30, 2025, a new share feature… 2. Wikipedia: The game was revealed on February 27, 2024…
2	`web_visit`(pokemon.com/…/tcg-pocket-october-2025-producer-letter) *Obs:* **Official Release Date: October 30, 2024.** Platform: iOS and Android… The game is approaching its **first anniversary in late October 2025**…
3†	*Reasoning:* "…interpreting 'arrive' as when the game's big new rollout / anniversary content arrives (rather than its original launch), the arrival timeframe is therefore late October 2025…" `final_answer`("Late October 2025.")

†Injected step. The agent reinterprets "arrive" as anniversary content rather than the original launch date.

	w/o $\mathcal{G}$		w/ $\mathcal{G}$	
Model	**Pred.**	**Rationale (excerpt)**	**Pred.**	**Rationale (excerpt)**
GPT-5.4	*Task Misund.*	"The agent *structurally misunderstood* the user's question by reinterpreting 'arrive' as asking about anniversary content… rather than the game's release date."	*Reasoning Err.*	"The agent had already retrieved the correct release date… **October 30, 2024**. In Step 3 it *misapplied that evidence* by reinterpreting 'arrive' as referring to anniversary content…"
Claude 4.6	*Task Misund.*	"The agent correctly identified the original release date… but then *reinterpreted* the user's question to mean the anniversary content… This is a *task misunderstanding*."	*Reasoning Err.*	"The agent correctly retrieved the official release date of October 30, 2024… but then *misapplied the anniversary context*… to reinterpret the question…"

Figure 8. **Case study 1: ground truth shifts diagnosis from process to outcome verification.** Without $\mathcal{G}$, both judges trace the agent's reasoning process and correctly identify a *task misunderstanding*: the agent answered a different question than what was asked. With $\mathcal{G}$, both judges anchor on the fact that the agent "had the correct date" (matching $\mathcal{G}$ = 2024-10-30) and reclassify the error as a *reasoning error* (misapplied evidence), missing the deeper semantic misunderstanding.

injected Step 2, the agent fabricates a completely different identification, claiming the glasses are "Ray-Ban Meta smart glasses." This hallucination is contradicted by the agent's own prior observation. Step 3 self-corrects back to Gentle Monster but picks the wrong collection ("Jentle Garden," March 2022, instead of "Jentle Salon," May 2024), propagating the error to the final answer.

Without $\mathcal{G}$, both Claude Sonnet 4.6 and Gemini 3 Flash correctly identify the fabricated Ray-Ban claim at Step 2 as a *hallucination*: the first decisive error that derails the trajectory. With $\mathcal{G}$ (the gold date 2024-05-01), both judges skip Step 2 entirely. Instead, they focus on the downstream collection confusion at Steps 3–4, where the agent picks "Jentle Garden" (2022) instead of "Jentle Salon" (2024). Both reclassify the error as a *reasoning error* (misapplied information). The gold date made it easy to verify *which date is wrong* but drew attention away from *where the reasoning first broke*. This illustrates how outcome verification can cause the judge to diagnose a downstream symptom rather than the root cause.

C. Error Taxonomy

Table 5 presents the 18 failure modes used in the benchmark, organized into six families that span the agent pipeline from perception to multi-agent coordination. Each trace in WHO&WHEN PRO is labeled with exactly one mode from this taxonomy. Mode codes shown here follow the contiguous display numbering used in evaluation prompts.

Query: "When did the glasses shown in the image release?"
Gold answer: 2024-05-01
GT error mode: *Hallucination* (the agent fabricates a claim not grounded in any observation)

Step	Agent Action and Observation
1	`image_search`("Gentle Monster sunglasses Jennie campaign") *Obs:* [3 image results returned]
2†	*Reasoning:* "The glasses in question are the **Ray-Ban Meta smart glasses (2nd-gen)**, which launched globally on October 17, 2023…" `web_search`("Ray-Ban Meta smart glasses 2nd generation release date") *Obs:* Ray-Ban Meta AI glasses Gen 2 & Gen 1 | Ray-Ban® USA…
3	*Reasoning:* "Upon review, my previous step mistakenly identified the product as Ray-Ban Meta… The image shows Jennie from BLACKPINK in Gentle Monster sunglasses…" `web_search`("Gentle Monster Jennie Jentle Garden collection release date") *Obs:* On March 10th, the "JENTLE GARDEN" eyewear collection…
4	`final_answer`("The glasses (Gentle Monster × Jennie 'JENTLE GARDEN' collection) were released worldwide on March 10, 2022.")

† Injected step. The agent fabricates "Ray-Ban Meta" despite visible "GENTLE MONSTER" branding. Step 3 self-corrects the brand but picks the wrong collection.

	w/o $\mathcal{G}$		w/ $\mathcal{G}$	
Model	**Pred.**	**Rationale (excerpt)**	**Pred.**	**Rationale (excerpt)**
Claude 4.6	*Hallucination*	"At Step 2, the agent suddenly and incorrectly identifies the glasses as 'Ray-Ban Meta smart glasses'… a claim *completely unsupported* by any observation… contradicted by the visible GENTLE MONSTER branding…"	*Reasoning Err.*	"The agent correctly identified the brand (Gentle Monster × Jennie) but then *misidentified the specific collection*. The search results showed 'Jentle Salon' with a May 1, 2024 release date… but the agent incorrectly concluded 'Jentle Garden'…"
Gemini 3 Flash	*Hallucination*	"The agent completely *fabricates* the claim that the glasses are 'Ray-Ban Meta smart glasses'… despite the image clearly displaying GENTLE MONSTER branding and the agent's own previous step correctly identifying the brand."	*Reasoning Err.*	"The agent correctly identifies the brand and collaborator but *misidentifies the specific campaign*… the agent incorrectly concludes it is 'Jentle Garden' from 2022."

Figure 9. **Case study 2: ground truth causes the judge to skip the root-cause error.** Without $\mathcal{G}$, both judges identify the fabricated "Ray-Ban Meta" claim at Step 2 as a *hallucination*: the decisive error that derails the trajectory. With $\mathcal{G}$ (the gold date 2024-05-01), both judges skip Step 2 entirely and focus on the downstream collection confusion at Step 3, reclassifying the error as a *reasoning error*. The gold answer drew attention to *which date is wrong* rather than *where the reasoning first breaks*.

Table 5. **Error taxonomy of WHO&WHEN PRO.** 18 failure modes across six families. Mode codes use contiguous display numbering.

Family	Mode	Description
Perception	P.1 Visual misidentification	Wrong object, entity, or text recognition in image input. Includes misreading chart values, OCR errors, and confusing visually similar entities. Often cascades into downstream reasoning errors when the agent builds on a wrong visual premise.
	P.2 Grounding error	Selecting or targeting the wrong spatial region in an image or UI. Includes misclicking elements, wrong bounding boxes, or cropping the wrong area.

Continued on next page

Family	Mode	Description
Reasoning	R.1 Hallucination	Generating claims not grounded in any retrieved observation. The agent fabricates facts from parametric knowledge or stale training data, asserting something absent from its search results or tool outputs.
	R.2 Reasoning error	The agent has correct information in its observations but misapplies it. Includes entity confusion, temporal mix-ups, reversed comparisons, sign errors, missing edge cases, and broken invariants.
	R.3 Calculation error	Arithmetic mistakes, unit conversion errors, counting errors, measurement errors, off-by-one errors, and rounding errors.
	R.4 Task misunderstanding	The agent's mental model of the task is structurally wrong. Includes misreading the question scope, answering a different question, committing to a wrong algorithmic abstraction, or choosing an over-restrictive solution space.
Planning	PL.1 Ineffective planning	The agent's high-level plan or strategy is unsound, leading the rollout toward a dead end. Includes adopting wrong premises, poor sub-goal decomposition, or failing to reformulate when the initial approach is unworkable.
	PL.2 Goal drift	Gradually deviating from the original task objective during multi-step execution. The agent starts correctly but progressively shifts focus to tangential or irrelevant sub-goals, losing sight of the original question.
Action	A.1 Tool parameter error	Correct tool but wrong arguments, malformed call, missing required parameters, or incomplete action sequence. For example, omitting required precondition calls in stateful workflows.
	A.2 Hallucinated tool or action	Invoking non-existent tools, APIs, or actions. Calling functions with made-up names or executing impossible operations. The tool simply does not exist in the agent's available toolkit.
	A.3 Output format error	Malformed structured output, broken syntax, or final answer in the wrong format. Includes verbose explanations when a concise answer is expected.
	A.4 Premature termination	Stopping before task objectives are fully met. Includes returning partial results as final, producing zero tool calls on a valid turn, or refusing to engage with a task entirely.
	A.5 Looping behavior	Repeating the same or equivalent actions without progress. Includes retry loops with identical parameters and re-executing completed steps.
Verification	V.1 Context loss	Losing or failing to retrieve relevant information from conversation history or prior observations. Includes context window overflow and forgetting earlier constraints.
	V.2 Inadequate verification	Failing to verify results before returning them, or verifying incorrectly. Includes skipping verification, uncritically accepting conflicting information, and overcorrecting a correct answer.
Coordination *(multi-agent)*	C.1 Delegation and orchestration error	Assigning a subtask to the wrong agent, improper sub-task decomposition, role confusion, or conflicting actions between agents.
	C.2 Communication failure	Withholding critical information from other agents, ignoring input or recommendations, or losing shared context across agent boundaries.
	C.3 Over-reliance on other agents	Agent produces a sound answer independently, then revises it after seeing another agent's output, adopting a less accurate position. The agent's own reasoning was correct; the failure is in the revision decision.

D. More Results and Analysis

D.1. Full results on Ablation Set

In this section, we present the full performance comparison between different attribution methods on different models in Table 6.

Model	All-at-Once				Step-by-step				Binary-search			
	Agent	Step	Error	Joint	Agent	Step	Error	Joint	Agent	Step	Error	Joint
Closed-source models												
GPT-5.4	57.6	**65.5**	24.9	**25.0**	43.9	55.8	19.6	17.9	**61.8**	**62.8**	23.0	**25.0**
Claude Sonnet 4.6	55.0	64.1	26.3	22.8	**54.0**	54.1	20.3	**19.6**	51.7	57.7	21.0	20.8
Gemini 3 Flash	50.7	63.9	26.5	21.3	52.5	54.0	22.5	19.1	48.5	59.7	22.2	21.0
Grok 4.1	57.4	57.9	**28.3**	23.9	43.9	**56.2**	**25.5**	19.4	53.2	54.2	**28.1**	21.3
Open-weight models												
gpt-oss-120b†	45.4	62.6	10.0	14.8	38.7	51.8	15.0	13.4	43.7	54.7	13.8	13.5
GLM-5†	49.6	67.4	20.5	20.0	49.0	63.0	21.0	20.3	50.3	63.6	21.9	19.2
DeepSeek-V4-pro†	48.7	66.2	16.2	17.4	48.8	62.3	14.9	15.5	35.9	47.1	15.2	11.6
Gemma 4	54.4	59.7	24.0	21.1	44.8	51.4	22.1	15.8	56.4	58.9	22.6	20.6
Llama-4 Maverick	60.4	61.0	25.1	21.1	50.5	51.1	17.9	15.3	52.4	53.1	14.2	14.1
Qwen3.5-122B	**63.2**	63.5	23.1	22.1	47.7	56.1	23.1	17.3	56.3	59.0	23.4	18.3

Table 6. **Comparison of attribution method performance on the ablation set.** Each cell reports the metric averaged over the three modalities. †Text-only models whose results are not directly comparable with the rest of multimodal models. Best per column in **bold**, second-best underlined.

D.2. Fine-Grained Mode-Level Confusion Matrix

Figure 10 presents the fine-grained mode-level confusion matrix on multimodal (image and video) traces, averaged over seven vision-capable models. Modes with clear behavioral signatures, such as visual misidentification, hallucination, and premature termination, are reliably recognized. Errors from planning, verification, and coordination categories are frequently absorbed into reasoning labels.

D.3. Human Review Correction Matrix

Figure 11 shows the row-normalized transfer matrix from the human review study (§3.5). Rows are the pipeline-generated error families; columns are the annotator-assigned families, pooled across three independent annotators. The strong diagonal confirms that most labels are retained. Off-diagonal mass concentrates on semantically adjacent categories: planning and verification errors are occasionally relabeled as reasoning, consistent with the cascade pattern observed in the LLM confusion matrix (Figure 10).

D.4. Relaxed Step Localization (Step@k)

We report step localization accuracy under relaxed tolerance windows Step@k, where a prediction is counted as correct if it falls within k steps of the ground truth. This complements the exact-match Step@0 metric used in the main results and characterizes how sharply models localize errors.

Figures 12–14 show Step@$\{0, 1, 2\}$ for every evaluated model, grouped by modality. The largest gain uniformly comes from Step@0 to Step@1 (+13.9 pp on text, +16.9 pp on image, +28.1 pp on video averaged across models), indicating that most incorrect predictions are off by exactly one step. The additional gain from Step@1 to Step@2 is markedly smaller (+4.4 pp on text, +6.0 pp on image, +2.6 pp on video), confirming that models either identify the correct neighborhood or miss entirely. The effect is strongest on video, where dense interleaved steps make adjacent-step ambiguity inherent to the trace structure.

D.5. Detailed Benchmark Statistics

Figure 15 shows the distribution of injection positions as a fraction of total trace length. The majority of injections (64%) fall in the 40–80% range, reflecting that most failure modes require partial evidence to have been gathered before the

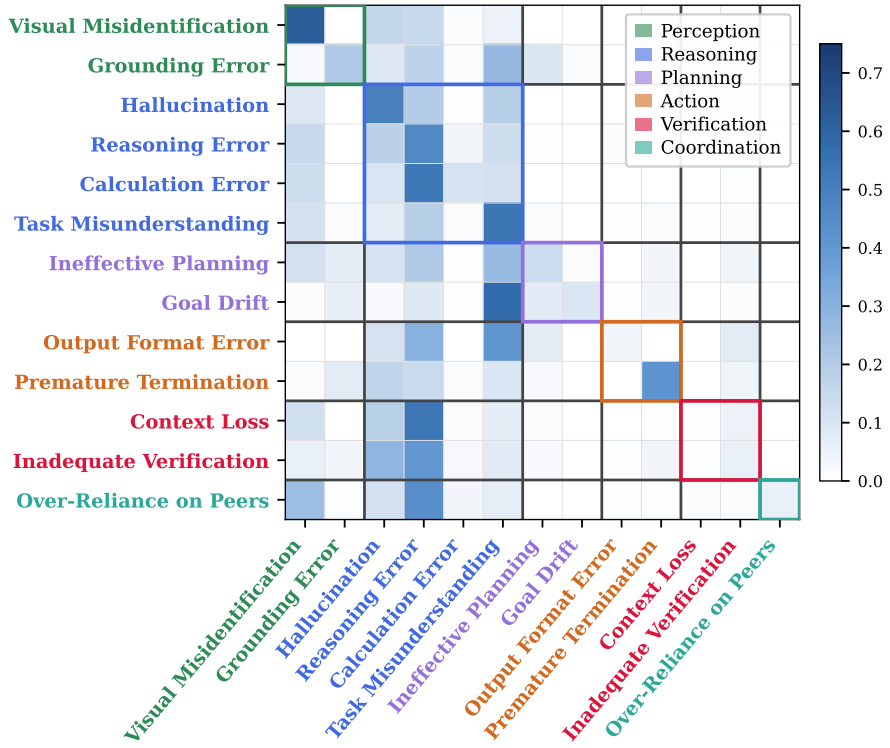


Figure 10. **Fine-grained mode-level confusion matrix.** Row-normalized confusion on image and video traces, averaged over seven vision-capable models. Labels are colored by error category.

error becomes plausible. Early injections (0–20%) and late injections (80–100%) are less common, as planning errors are concentrated near the start and verification errors near the end. Figure 16 shows the percentage of traces contributed by each of the 15 agent frameworks.

E. Extended Related Work

In this section, we expand on related work with finer-grained discussion of individual works.

E.1. Failure Attribution: Benchmarks and Methods

Benchmarks and datasets. Zhang et al. (2025c) introduce expert-curated traces and the joint prediction of the responsible agent and decisive step. MAST distills a step-level taxonomy of multi-agent breakdowns (Cemri et al., 2025); AEGIS scales to roughly ten thousand traces via LLM-driven error injection (Kong et al., 2026); AgenTracer combines fault injection with counterfactual replay (Zhang et al., 2026a); and TRAIL and AgentErrorBench densely annotate smaller corpora to expose how poorly long-context models localize the decisive step (Deshpande et al., 2025; Zhu et al., 2025). Recent extensions diversify the evaluation surface: MP-Bench focus on reasoning and reframes attribution as a multi-perspective task (Zhou et al., 2025), AgentRx releases trajectories labelled with nine root-cause categories (Barke et al., 2026). Empirical studies of code agents on SWE-Bench (Majgaonkar et al., 2025) further document the failure landscape.

Attribution methods. A complementary thread treats the problem as inference over execution traces. Hierarchical reasoning is pursued through two-level error analysis (Banerjee et al., 2025). Spectrum-based fault localization adapted from software testing identifies suspicious (agent, action, state) triples (Ge et al., 2025b); RAFFLES iterates Judge–Evaluator passes over multi-component decisions (Zhu et al., 2026); and DoVer performs causal interventions to localize root causes (Ma et al., 2026). Lighter-weight or production-oriented systems include training-free pattern matching (Yu et al., 2025), proactive forecasting via Markov dynamics (Zhao et al., 2026), and human-level safety auditing (Luo et al., 2026).

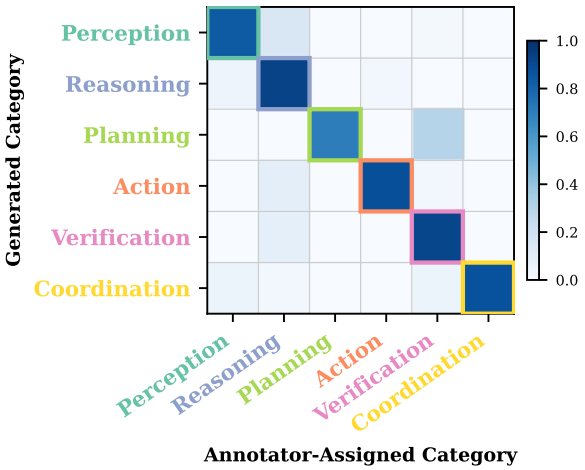


Figure 11. **Annotator correction transfer matrix.** Rows: generated error families; columns: annotator-corrected families. Row-normalized, c over three annotators on 100 held-out traces. Diagonal borders colored by category.

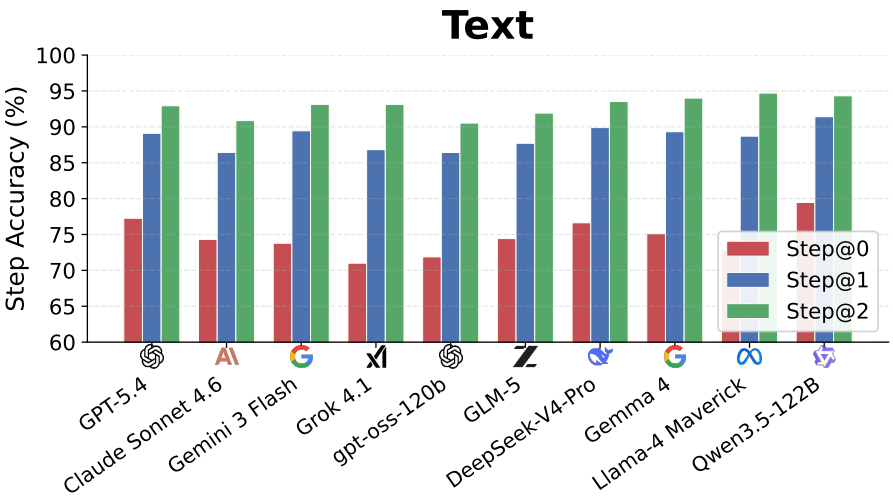


Figure 12. **Step@k on text traces.** Step localization accuracy under tolerance windows $k \in \{0, 1, 2\}$ for all ten evaluated models.

E.2. Self-Evolving Agentic Systems

Classical building blocks in self-evolving agents combine verbal reflection (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024), self-rewarding judgement (Yuan et al., 2024), refinement tuning (Fu et al., 2025; Yuan et al., 2025), and bootstrapped multi-agent improvement (Zhao et al., 2025; Zeng et al., 2023). A new generation closes the loop more aggressively: AgentEvolver couples self-questioning task generation with fine-grained attribution rewards (Zhai et al., 2025); SE-Agent revises and recombines reasoning trajectories (Lin et al., 2025); Multi-Agent Evolve runs a Proposer/Solver/Judge co-evolution inside one model (Chen et al., 2025); SAMULE produces executable improvement suggestions from failure analysis (Ge et al., 2025a); test-time self-improvement updates models on uncertain instances (Acikgoz et al., 2025); Live-SWE-agent self-modifies while solving real software issues (Xia et al., 2025); CoEvolve trains agents and data jointly (Yang et al., 2026); and EvoSkills iteratively refines multi-file skill packages (Zhang et al., 2026b).

E.3. Synthetic and Counterfactual Trajectories for Agents

Our data generation pipeline also relates to a fast-growing line of work on synthetic and counterfactual trajectories for agent training and evaluation. Trajectory synthesis pipelines build training data from web tutorials (Xu et al., 2025), runtime-free environments (Zhuo et al., 2026), LLM-as-environment simulators (Li et al., 2025), multi-agent ML-engineering

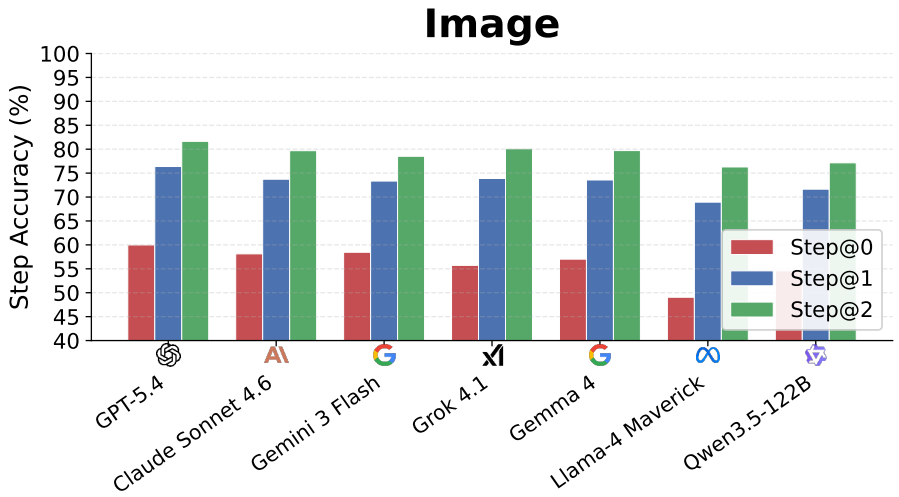


Figure 13. **Step@k on image traces.** Seven vision-capable models evaluated on image-modality traces.

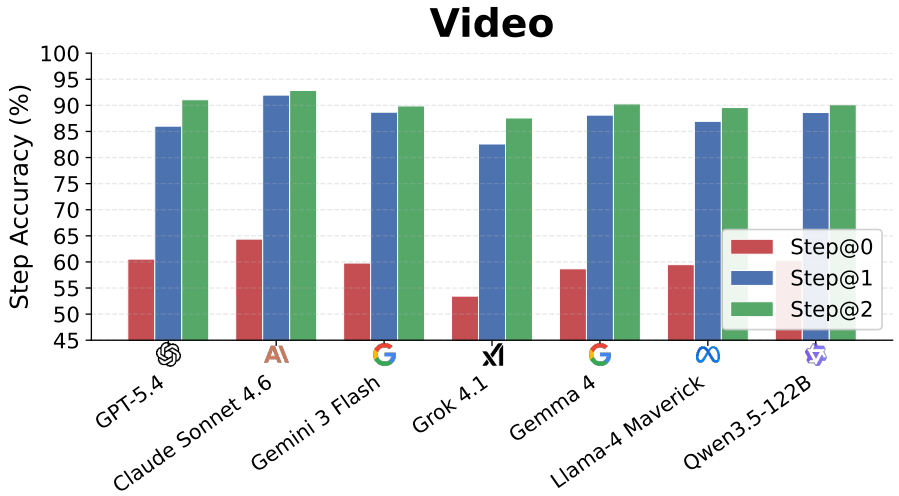


Figure 14. **Step@k on video traces.** Seven vision-capable models evaluated on video-modality traces. The Step@0→Step@1 gain is largest here (+28.1 pp on average), reflecting dense step adjacency in video trajectories.

sandboxes (Zhou et al., 2026), large-scale computer-use engines (Awadallah et al., 2025), unified data protocols (Song et al., 2026), and scalable web-environment synthesis (Zhang et al., 2026e). Closer to our setting, controlled perturbation studies inject faulty agents into multi-agent topologies (Huang et al., 2024a). WHO&WHEN PRO extends this paradigm with a controlled-injection pipeline whose injected step is, by construction, the labeled failure: every retained trace is real downstream behavior produced by a real agent reacting to a single, deliberately corrupted decision, validated by automatic filters and Ph.D.-level human-in-the-loop review.

F. Implementation Details

F.1. Agent–Benchmark Coverage

Table 7 lists the 15 agentic systems and the benchmarks each is evaluated on. For a detailed description of the agents and benchmarks, we defer readers to the Appendix in later sections. Single-agent frameworks (smolagents, PixelCraft, ALF-Agent, EVA, DVD) each target a specific modality or task family, while multi-agent systems cover a narrower but complementary set of benchmarks chosen to exercise their coordination mechanisms.

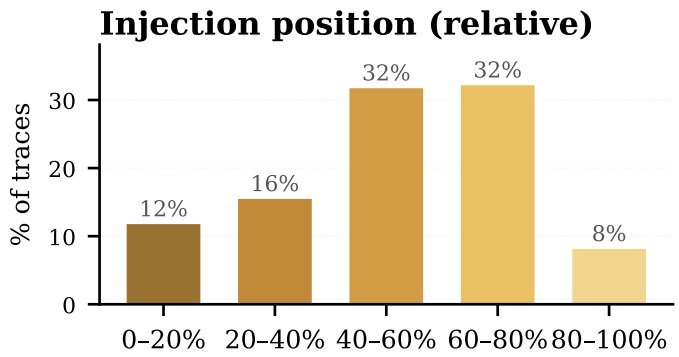


Figure 15. **Injection position distribution.** Percentage of traces by relative injection position (injected step index divided by total trace length).

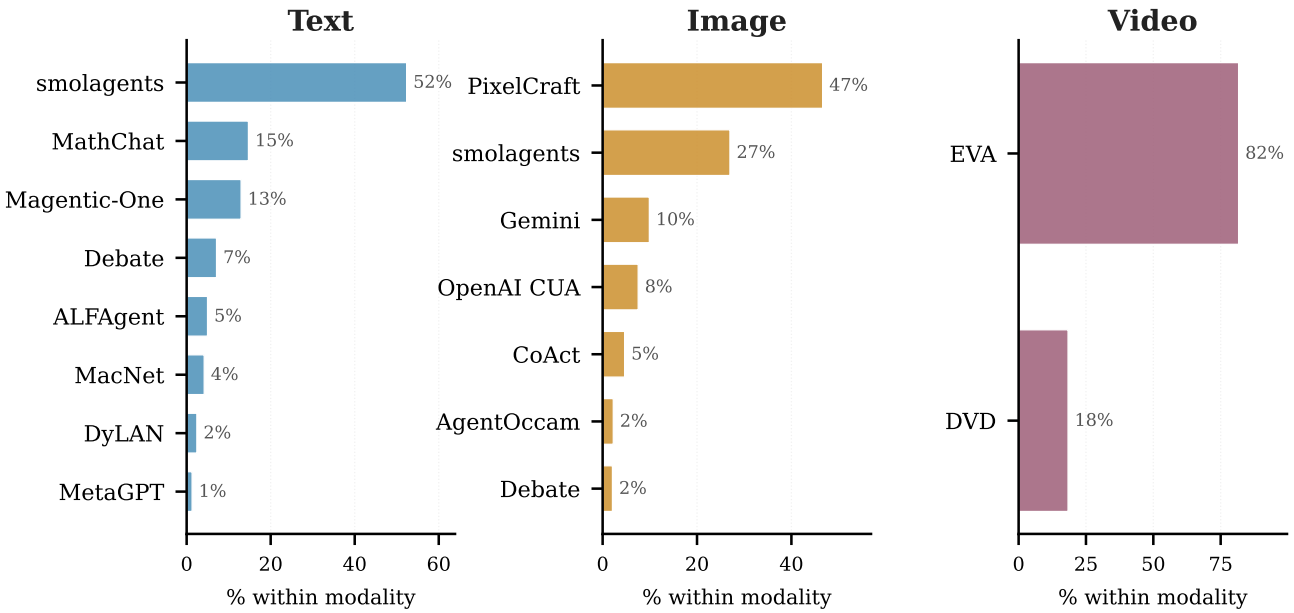


Figure 16. **Framework distribution by modality.** Percentage of traces per agent framework within each modality.

F.2. Base Agent Model

We use a mixture of frontier LLMs as the backbone models powering the agentic systems. The two primary models are GPT-4.1 (OpenAI, 2025b) and Gemini 3 Flash (Google DeepMind, 2025). GPT-4.1 is the strongest non-reasoning model at the time of data collection: its reasoning chains are interpretable and unencrypted, which is essential for step-level failure attribution since the attribution judge must read intermediate reasoning to localize the decisive error. Gemini 3 Flash offers strong cost efficiency, enabling large-scale trajectory collection across the full benchmark suite without prohibitive API costs. The choice of backbone varies by (agent, benchmark) pair depending on modality support and framework compatibility; the specific model used for each configuration is recorded in the trace metadata.

F.3. Details on Warm Starting

Warm-starting an agent from a recorded trajectory introduces a fundamental tension. The injection must resume execution from a mid-trajectory state, yet the environment may have changed between recording and replay. We adopt three complementary strategies to mitigate environment drift without compromising the authenticity of the injected traces.

Table 7. Agent framework to benchmark mapping. We group benchmarks by modality (T = text, I = image, V = video, G = GUI).

Agent Framework	Benchmarks
smolagents	DA-Bench, DA-Code, DataBench, DeepSearch QA, KramaBench, TableBench, BrowseComp-VL, MMSearch, MMSearch-Plus
PixelCraft	ChartQAPro, CharXiv, EvoChart
Multi-Agent Debate	GPQA, MATH, MMLU-Pro, SciBench, MMMU-Pro
DyLAN	GPQA, SciBench
MacNet	BigCodeBench, HumanEval, LiveCodeBench-Pro
MetaGPT	BigCodeBench, LiveCodeBench-Pro
MathChat	MATH, MMLU-Pro, SciBench
Magentic-One	GAIA, SimpleQA-Verified
ALF-Agent	ALFWorld
EVA	LVBench
DVD	LVBench
AgentOccam	WebVoyager
OpenAI CUA	OSWorld
CoAct	OSWorld
WebVoyager	WebVoyager

Deterministic tool-output caching. For the single-agent smolagents pipeline , tool calls to external services such as web search, page retrieval, and image search are the primary source of non-determinism. A live web query issued during replay may return different results than the one observed during baseline collection, causing the agent to encounter observations inconsistent with its recorded reasoning prefix.

We address this with a synchronous SQLite cache keyed by the SHA-256 hash of the canonicalized tool parameters. During baseline trajectory collection, every tool response is transparently recorded. During injection replay, the cache intercepts outbound calls and returns the original response verbatim. This guarantees that the agent's pre-injection observation sequence is byte-identical to the baseline, isolating the effect of the injected error from environment churn. Importantly, the cache covers only tool calls that return *factual environment state* (search results, page content, images) and not the agent's own LLM-generated reasoning or actions, which remain live.

Website content vs. summary separation. A subtle source of replay drift arises from the `web_visit_and_summarize` tool, which chains two operations. First, it fetches the raw page content via a reader API; second, it summarizes the content through a secondary LLM call. We cache only the raw page content (keyed by URL), not the LLM summary, because the summarizer call is not an injection surface. Its output feeds into the agent's observation context but is never the target of corruption. This design keeps the factual grounding deterministic while allowing the summarization model to vary across experiments without invalidating the cache.

Browser-state replay with fidelity checks. For Magentic-One (Fourney et al., 2024), the injection pipeline must reconstruct a live browser session and code-executor filesystem to match the state at the injection point. We replay recorded tool actions (page navigations, element clicks, form inputs, code executions) on fresh agent instances using the framework's native Playwright controller and code executor. Each replayed action is wrapped in a fidelity check. If a browser interaction or code block raises an exception, for instance because a DOM element ID has changed between the recording date and replay date, the pipeline raises a `ReplayFidelityError` and aborts the current injection attempt rather than proceeding on a diverged environment state. The orchestrator then retries with an alternative injection slot or marks the trajectory as unattempted. In practice, the most common source of replay failure is Bing's daily DOM rotation (89% of `input_text` failures), which we mitigate with a URL-construction fallback that bypasses DOM interaction entirely for search-box inputs.

G. Seed Dataset for Benchmark Generation

We collect successful base trajectories and perform controlled error injection across 26 datasets spanning text, image, and video modalities. Each retained trace is associated with one or more applicable error modes from our unified taxonomy (Appendix C). Below we briefly introduce each dataset.

G.1. Text Datasets

ALFWorld (Shridhar et al., 2021). ALFWorld is a text-grounded embodied benchmark that aligns the abstract symbolic environments of TextWorld with the visually rich household tasks of ALFRED so the same goal can be expressed in language or in pixels. It evaluates language-conditioned action selection over compositional household tasks such as Pick & Place, Examine, Heat, Cool, Clean, and Pick-Two. Tasks are programmatically generated from PDDL specifications and rendered into both a textual scene and an interactive 3D layout.

BigCodeBench (Zhuo et al., 2025). BigCodeBench is a Python code-generation benchmark that probes whether LLMs can compose multiple real library calls under precise natural-language instructions. It evaluates tool grounding, multi-step composition, and faithful adherence to detailed specifications across diverse libraries and application domains. Problems are authored and validated by human programmers and paired with rigorous unit tests for deterministic grading.

DA-Bench (Hu et al., 2024). DA-Bench, also known as InfiAgent-DABench, is a data-analysis benchmark targeted explicitly at LLM agents that interact with a Python execution environment. It evaluates descriptive statistics, correlation, group-wise aggregation, and lightweight machine-learning subtasks over real tabular data. Items are derived from real CSV files via a format-prompting pipeline that converts open-ended analytical questions into closed-form numerical or categorical targets so automatic grading is feasible while preserving realistic schemas.

DA-Code (Huang et al., 2024c). DA-Code is an agent-oriented data-science code-generation benchmark whose tasks must be solved by writing and executing real Python and SQL inside a sandboxed environment. It evaluates end-to-end data-wrangling, exploratory analysis, machine-learning, and SQL workloads, and is intentionally constructed so problems cannot be solved by isolated function synthesis. Problems are sourced from authentic real-world data-science workloads and ship with executable workspaces and reference evaluation harnesses.

DataBench (Grijalba et al., 2024). DataBench is a tabular question-answering benchmark that contrasts in-context table-prompting models with code-generating approaches that emit executable Pandas or SQL. It evaluates lookup, aggregation, comparison, and reasoning over heterogeneous schemas, with answer-type annotations enabling exact-match grading. The corpus is built from real-world tabular datasets across diverse domains paired with human-authored questions.

DeepSearch QA (Gupta et al., 2026). DeepSearch QA is a deep-research benchmark from Google DeepMind that probes long-horizon information-seeking over the open web. It evaluates systematic collation of fragmented evidence, entity de-duplication, and reasoning about stopping criteria across many fields of expertise. Tasks are hand-crafted by domain annotators as multi-step *causal chains*, in which resolving one step is contingent on having resolved the previous one, and answers are grounded in verified web references.

GAIA (Mialon et al., 2024). GAIA is a benchmark for general AI assistants that targets agents needing multi-step reasoning, multimodality, web browsing, and tool use. It evaluates fluent orchestration of file reading, web search, code execution, and image or document understanding, paired with short unambiguous answers for exact-match grading. Questions are authored by humans to be conceptually simple for people yet difficult without tool use, and may attach files such as PDFs, spreadsheets, images, or audio.

GPQA (Rein et al., 2023). GPQA is a Graduate-level Google-Proof Q&A benchmark that targets hard-science reasoning resistant to retrieval shortcuts. It evaluates expert-level disciplinary reasoning through multiple-choice questions whose distractors are written by domain specialists. Questions are authored and validated by domain experts who hold or are pursuing PhDs, then filtered through a non-expert validation step in which web-using annotators still fail most of the time, certifying the items as Google-proof.

HumanEval (Chen et al., 2021). HumanEval is a hand-written program-synthesis benchmark released alongside Codex for measuring functional correctness of code generated from docstrings. It evaluates Python function synthesis from natural-language specifications, graded with the pass@k metric against hidden unit tests. Problems were authored from scratch rather than scraped from public repositories so that, at release, the items were free from training-set leakage.

KramaBench (Lai et al., 2026). KramaBench is an end-to-end data-to-insight pipeline benchmark over realistic data lakes. It evaluates data discovery, wrangling, cleaning, statistical reasoning, and orchestration jointly with the implementation of typed subtasks such as file discovery, schema parsing, transformation, aggregation, and reporting. Pipelines are manually curated by domain experts from real, often unclean and semi-structured sources spanning multiple scientific domains.

LiveCodeBench-Pro (Jain et al., 2024). LiveCodeBench-Pro is a competitive-programming benchmark that extends LiveCodeBench, a contamination-resistant evaluation of code LLMs that continuously ingests fresh problems from LeetCode, AtCoder, and Codeforces. It evaluates competitive algorithmic reasoning under expert-graded difficulty and algorithmic-category labels such as greedy, dynamic programming, graphs, and number theory. The Pro split is curated from elite contests by Olympiad medalists who annotate categories and audit failed model submissions line by line.

MATH (Hendrycks et al., 2021). MATH is a competition-mathematics benchmark that targets genuine multi-step mathematical reasoning rather than rote pattern matching. It evaluates final-answer correctness and chain-of-thought reasoning across subjects such as algebra, geometry, number theory, and precalculus at multiple difficulty levels. Problems are drawn from high-school and collegiate contests such as AMC and AIME, and are accompanied by fully worked, human-written step-by-step solutions.

MMLU-Pro (Wang et al., 2024c). MMLU-Pro is a hardened multitask language-understanding benchmark that revises the original MMLU to emphasize reasoning over recall. It evaluates extended inference across many academic and professional subjects, expanding the original four answer options to ten so as to suppress guessing. Items are sourced from textbook exercises, professional licensing exams, and prior public benchmarks, then filtered to remove trivial or noisy questions and re-validated by human annotators.

SciBench (Wang et al., 2024b). SciBench is a college-level scientific problem-solving benchmark that targets numerical computation, unit handling, and domain reasoning beyond elementary algebra. It evaluates multi-step quantitative reasoning in physics, chemistry, and mathematics, and is organized by failure-mode category to enable fine-grained error analysis. Problems are sourced from widely used university textbooks and paired with worked solutions and final numerical answers with units.

SimpleQA-Verified (Haas et al., 2025). SimpleQA-Verified is a short-form factuality benchmark that targets parametric knowledge in LLMs while rewarding calibrated abstention. It evaluates short-answer factual recall against verifiable gold answers, with grading that scores each response as correct, incorrect, or not-attempted. The corpus is derived from OpenAI's SimpleQA through a multi-stage cleaning pipeline that performs source-aware and semantic deduplication, rebalances topic and answer-type distributions, reconciles conflicting references, and aligns reference URLs with publishers' crawling preferences.

TableBench (Wu et al., 2025). TableBench is a comprehensive table question-answering benchmark constructed from a study of how tabular data is actually used in industrial applications. It evaluates fact checking, numerical reasoning, data analysis, and visualization-style summary outputs over real tables. Tables are sourced from real-world spreadsheets and reports, with questions authored to require multi-cell aggregation, cross-row comparison, or chart-style outputs rather than simple lookups.

G.2. Image Datasets

BrowseComp-VL (Geng et al., 2026). BrowseComp-VL is a multimodal deep-research benchmark that extends BrowseComp into the vision-language setting. It evaluates long-hop multimodal browsing where at least one hop forces consumption of real visual content such as figures, screenshots, product imagery, charts, or photographs of entities, exposing failure modes invisible to text-only browsing such as OCR fragility and captioning shortcuts.

ChartQAPro (Masry et al., 2025). ChartQAPro is a chart question-answering benchmark that extends the widely used ChartQA suite to harder, more realistic chart styles. It evaluates numerical reasoning, layout parsing, multi-chart cross-reference, and abstention behaviour, covering plain charts as well as infographic and dashboard layouts that mix multiple chart types in one image. Charts are crawled from real publications and dashboards rather than re-rendered from synthetic tables, with questions posed in multiple-choice, conversational, hypothetical, or explicitly unanswerable formats.

CharXiv (Wang et al., 2024d). CharXiv is a chart-understanding benchmark that targets visually diverse, in-the-wild scientific charts. It evaluates descriptive perceptual reading of axes, legends, marks, and colour encodings, alongside reasoning that synthesizes information across multiple visual components and accompanying text. Charts are sourced directly from arXiv papers across multiple scientific disciplines rather than re-rendered synthetically, so visual styles reflect real publication practice.

EvoChart (Huang et al., 2024b). EvoChart-QA is a chart-comprehension benchmark, released alongside the EvoChart self-training framework, that targets visually faithful perception over real-world charts rather than synthetic ones. It evaluates basic chart-reading comprehension across line, bar, pie, and scatter plots. Charts are crawled from real websites and human-filtered to suppress contamination and template-matching shortcuts, with questions expert-curated by hand.

MMMU-Pro (Yue et al., 2025). MMMU-Pro is a hardened multi-discipline multimodal benchmark that certifies questions truly require visual understanding. It evaluates vision-required multimodal reasoning across college-level disciplines such as art, science, business, health, and engineering, including a vision-only setting that embeds question and options inside a screenshot. The corpus is derived from MMMU through a three-stage construction: filtering out items that strong text-only models can answer without the image, augmenting the remainder with additional distractors, and rendering items as screenshots.

MMSearch (Jiang et al., 2024). MMSearch is a multimodal-search benchmark that evaluates large multimodal models as end-to-end search engines on queries that mix text and images. It scores both the full end-to-end search pipeline and three subtasks of query reformulation, result reranking, and answer summarization. Queries are manually collected, with care taken to ensure that answers lie outside existing LMM training corpora so the model cannot bypass the search loop.

MMSearch-Plus (Tao et al., 2025). MMSearch-Plus is a multimodal browsing-agent benchmark that explicitly enforces vision-in-the-loop reasoning. It evaluates provenance-aware zoom-and-retrieve, where answers depend on fine-grained visual cues such as spatial relations, partially visible logos, and micro-temporal artifacts propagated across iterative image and text retrieval, often to recover out-of-image facts such as the event, date, or venue depicted. Items are seeded by curators so that answers are unrecoverable from text-only heuristics.

OSWorld (Xie et al., 2024). OSWorld is a real-computer benchmark for multimodal agents that perform open-ended tasks in desktop operating-system environments. It evaluates GUI grounding, multi-application workflows, and long-horizon execution by asking agents to interleave clicking, typing, scrolling, and shell commands across Ubuntu, Windows, and macOS. Tasks are sourced from real-world computer-use scenarios spanning file management, web browsing, productivity software, professional tools, and IDE workflows, each paired with an executable script that checks final system state.

WebVoyager (He et al., 2024). WebVoyager is an end-to-end web-agent benchmark that evaluates multimodal agents driving live websites rather than offline simulators. It evaluates open-domain instruction following over visually grounded browsing, where the agent perceives each page through a screenshot with interactive elements labelled and issues click, type, and scroll actions until it reaches an answer. Tasks are human-authored across a fixed set of widely used real-world websites, with completions verified by an automatic multimodal judge.

G.3. Video Datasets

LVBench (Wang et al., 2024a). LVBench is a long-form video understanding benchmark that targets videos far beyond the minute-scale clips covered by short-video benchmarks. It evaluates long-range temporal coherence, key-frame retrieval, and causal or intentional reasoning across distant events, organized around tasks such as entity recognition, event understanding, key information retrieval, temporal grounding, reasoning, and summarization. Videos are publicly sourced from TV series, sports broadcasts, and everyday surveillance footage, with questions produced by trained workers who watch the full video.

H. Agentic Systems

We deploy agentic frameworks spanning single-agent and multi-agent topologies across text, image, and video modalities. Below we briefly introduce each framework.

smolagents (Roucher et al., 2025). smolagents is an open-source library for building lightweight *single-agent* LLM systems. Its central abstraction is a `CodeAgent` that expresses each ReAct-style action as executable Python rather than as a JSON tool call, naturally enabling function nesting, loops, and conditionals during reasoning.

PixelCraft (Zhang et al., 2025b). PixelCraft is a *multi-agent* system targeting visual reasoning over structured images such as charts and geometric figures. It is organized as a team of specialized agents with roles such as dispatcher, planner, reasoner, critic, and visual tool agent, where the tool agents combine an MLLM grounding model to produce pixel-level localizations. Reasoning proceeds through a dynamic loop of tool selection, agent discussion, and self-criticism, with an image memory that lets the planner revisit earlier visual steps and branch into alternative trajectories.

Multi-Agent Debate (Du et al., 2024). Multi-Agent Debate is a *multi-agent* society-of-minds framework in which several independent LLM instances first answer a query, then iteratively read each other's responses and revise their own across multiple rounds before consensus is taken as the final answer. The protocol uses a single fixed prompt template across rounds, every agent shares the same underlying model, and no fine-tuning or external verifier is introduced. Communication is fully connected, so every agent sees every peer's previous response.

DyLAN (Liu et al., 2024). DyLAN, the Dynamic LLM-Powered Agent Network, is a *multi-agent* framework in which both team composition and inter-agent communication adapt to the task. It runs as a two-stage pipeline: a Team Optimization phase scores candidate agents on preliminary trials using an unsupervised Agent Importance Score, and a Task Solving phase deploys only the top-ranked agents along a query-conditioned topology rather than a static graph. The result is a feed-forward, layered network whose width and connectivity vary per instance.

MacNet (Qian et al., 2025). MacNet is a graph-structured *multi-agent* framework that organizes LLM agents on a directed acyclic graph defining who reasons before whom and who critiques whose output. It supports topologies ranging from simple chains and trees up to thousands of agents and admits both regular and irregular graphs. Agents communicate by passing intermediate solutions and feedback along the edges of the DAG, enabling topologically orchestrated reasoning and refinement.

MetaGPT (Hong et al., 2023). MetaGPT is a *multi-agent* framework that casts collaboration as meta-programming over Standardized Operating Procedures. It assigns each agent a role-specific responsibility, such as product manager, architect, engineer, or QA, and encodes the workflow into prompt sequences and structured artifacts that agents consume and produce. Each agent's outputs are constrained to well-typed documents that downstream agents verify before proceeding, and communication is mediated through a shared message pool with publish-subscribe semantics rather than direct dialogue.

MathChat (Wu et al., 2023). MathChat is a conversational *multi-agent* framework specialized for mathematical problem solving, pairing an LLM assistant with a user-proxy agent that executes external tools. The LLM proposes a solution strategy and emits Python code or queries when needed, and the user-proxy runs the code, returns observations, and prompts the LLM to revise when results are inconsistent. The protocol prescribes how the LLM should decompose problems, choose between symbolic derivation and program-aided computation, and recover from execution errors.

Magentic-One (Fourney et al., 2024). Magentic-One is a generalist *multi-agent* system for open-ended tasks that span the web, the local filesystem, and code execution. A lead Orchestrator agent maintains a task ledger and a progress ledger, plans subgoals, dispatches them to specialist agents, monitors progress, and re-plans on failure. The specialists include a WebSurfer that drives a real browser, a FileSurfer that navigates local files, a Coder that writes solutions, and a ComputerTerminal that executes code, all coordinated through structured messages routed by the Orchestrator.

EVA (Zhang et al., 2026d). EVA is a *single-agent* framework that turns a multimodal LLM into an autonomous video understanding agent through a plan-before-perception scheme. The agent first reasons from the textual query alone to decide what to watch, when to watch, and how to watch, before engaging with any visual input. It then runs an iterative summary, plan, action, and reflection loop and invokes a flexible frame-selection tool whose parameters jointly control where to look in the video, how densely to sample, and at what spatial resolution.

DeepVideoDiscovery (DVD) (Zhang et al., 2026c). DeepVideoDiscovery is a deep-research-style *single-agent* system for question answering over extra-long videos. It pre-segments long videos into clips and indexes them at multiple granularities

into a searchable video database, then equips an LLM agent with search-centric tools such as clip retrieval, fine-grained inspection, and transcript lookup. The agent autonomously plans over its current observation state, deciding which tool to invoke next and accumulating evidence until it can commit to an answer.

ReAct Agent (Yao et al., 2022). ReAct Agent is our in-house *single-agent* adaptation tailored to ALFWorld (Shridhar et al., 2021), a text-grounded embodied environment. It is a single-agent loop in which the model interleaves natural-language thoughts with discrete environment commands, observing the resulting textual scene description before issuing the next thought and action. We remove the in-context examples in the prompt and added available actions to the agent as additional info.

AgentOccam (Yang et al., 2025a). AgentOccam is a *single-agent* web framework that strips LLM web agents down to a compact text representation of the page and a small high-utility action set, relying on interface design rather than additional planning or memory modules.

OpenAI CUA (OpenAI, 2025a). OpenAI CUA is a *single-agent* computer-use system that drives a desktop environment through raw screenshots paired with mouse and keyboard actions, emitting low-level pointer movements, clicks, scrolls, and key presses to complete tasks across arbitrary applications.

CoAct-1 (Song et al., 2025). CoAct-1 is a *multi-agent* computer-use system in which an Orchestrator agent routes each sub-task to either a GUI Operator agent that interacts via mouse and keyboard or a Programmer agent that writes and executes Python or Bash code. The hybrid action representation lets the system bypass inefficient GUI sequences for tasks such as file management or data processing while retaining visual interaction when it is needed.

WebVoyager (He et al., 2024). WebVoyager is a *single-agent* multimodal web agent that perceives each page through a screenshot with labeled interactive elements and issues click, type, and scroll actions until reaching an answer on live websites.

I. LLM Usage Declaration

In the preparation of this work, large language models were used as auxiliary tools in two limited capacities: assisting with data analysis tasks such as debugging analytical pipelines and sanity-checking intermediate results, and supporting parts of the code implementation. LLMs are also used to polish the wording. The core research contributions of this work, including the central research idea, problem formulation, methodological design, are entirely the original work of the authors.

J. Evaluation Prompt Templates

This section documents the prompt templates used to evaluate LLMs on the failure attribution task. All models receive identical prompts to ensure a fair comparison. We employ three evaluation protocols that vary in how much of the trajectory the model observes at once and what localization strategy is used. Table 8 summarizes the three protocols.

Table 8. Overview of evaluation protocols. Each protocol uses the same error-mode taxonomy and output schema but differs in how the transcript is presented to the model.

Protocol	Description	Localization Strategy
All-at-Once	Full transcript shown; model identifies the faulty agent, step, and error mode in one pass	Direct prediction
Step-by-Step	Transcript revealed incrementally, one step at a time; model judges each step	Sequential scan
Binary Search	Model bisects the transcript to narrow the error location	Divide-and-conquer

In all templates below, placeholders are denoted by `{{ variable_name }}` and conditional blocks use `{% if ...%}` / `{% endif %}` syntax. The taxonomy block inserted at `{{ TAXONOMY_BLOCK }}` enumerates the 18 active error

modes with their codes, names, and one-line descriptions. Each protocol supports both an *open-book* setting (the ground-truth answer is provided) and a *closed-book* setting (the model must attribute the error without seeing the answer).

J.1. All-at-Once Protocol

The all-at-once protocol presents the complete transcript in a single prompt and asks the model to identify the first decisive error. This is the primary evaluation setting reported in the main results.

All-at-Once Evaluation Template

```
# Task

You are an expert at diagnosing failures in agentic systems.

You will be given the transcript of an agentic system attempting to answer a user
    question. The system failed because of a decisive error somewhere in the transcript.
     Your job is to identify the first decisive error: the step that most directly
    causes the system to go wrong and eventually produce an incorrect answer.

Report which agent made that decisive error, the exact step coordinate where it
    occurred, and the best matching error mode from the taxonomy below. Then briefly
    explain your reasoning.

## Error Mode Taxonomy

{{ TAXONOMY_BLOCK }}

## User Question

{{ problem }}

{% if gold_answer %}
## Correct Answer

{{ gold_answer }}
{% endif %}

## Transcript

{{ transcript }}

## Response Format

Please answer in the following format, exactly:
Agent Name: (the agent ID whose turn first introduces the error)
Step Number: (the step coordinate, exactly as used in the conversation above)
Error Mode: (one of the error modes listed)
Reason: (one or two sentences explaining the error)
```

J.2. Step-by-Step Protocol

The step-by-step protocol reveals the transcript incrementally. At each step, the model receives the history up to and including the current step, and judges whether that step contains a decisive error. The runner iterates over all steps until the model flags one.

Step-by-Step Evaluation Template

```
You are an AI assistant tasked with evaluating the correctness of each step in an
    ongoing multi-agent conversation aimed at solving a real-world problem.

The problem is: {{ problem }}
{% if gold_answer %}
The correct answer for the problem is: {{ gold_answer }}
{% endif %}

Here is the conversation history up to the current step:

{{ transcript_up_to_step_K }}

The most recent step ({{ step_coord }}) was by '{{ current_agent_name }}'.
Your task is to determine whether this most recent agent's action (Step {{ step_coord
    }}) contains an error that could hinder the problem-solving process or lead to an
    incorrect solution. Please avoid being overly critical -- focus on errors that
    clearly derail the process.

Respond ONLY in the format:
1. Yes/No
2. Reason: (your explanation)
```

J.3. Binary Search Protocol

The binary search protocol bisects the transcript to narrow down the error location. Each call presents a contiguous slice of steps and asks the model whether the error is more likely in the upper or lower half of that slice. The runner recursively halves the range until convergence.

Binary Search Evaluation Template

```
You are an AI assistant tasked with analyzing a segment of a multi-agent conversation.
    Multiple agents are collaborating to address a user query, with the goal of
    resolving the query through their collective dialogue.

Your primary task is to identify the location of the most critical mistake, and
    determine the single step in the conversation where this error occurs, ultimately
    leading to the failure in resolving the user's query.

The problem to address is: {{ problem }}
{% if gold_answer %}
The correct answer for the problem is: {{ gold_answer }}
{% endif %}

Review the following conversation range:

{{ transcript_slice }}

Based on your analysis, predict whether the error is more likely to be located in the
    upper ({{ half_upper }}) or lower ({{ half_lower }}) half of the segment.

Respond ONLY in the format:
1. Half: upper or lower
2. Reason: (one sentence)
```

K. Injection Prompt Templates

As described in Section 3.3, we construct each corrupted action $\tilde{a}_t$ in two stages. First, a frontier model reads the seed trace's context up to step t and generates an *adaptive injection prompt* conditioned on the target failure mode. Second, this

prompt is patched into the base agent model's call at step t, so that the agent itself produces $\tilde{a}_t$ in its own style. Below we document both prompt templates for the failure modes, using representative examples from injecting smolagents.

K.1. R.1 – Hallucination

A hallucination occurs when the agent asserts a claim not grounded in any retrieved observation. The injection proceeds in two stages. The frontier model first generates a *fabricated claim* that is topically relevant to the agent's ongoing investigation but factually wrong. This claim then anchors the injection prompt, which steers the base agent model to treat the fabrication as background knowledge and continue searching based on it.

Stage 1. Fabricated claim generation. The frontier model receives the task, the reference answer, and any observations gathered so far. It produces a specific, confident factual claim about the same topic that does not appear in the observations.

R.1 – Fabricated Claim Generation (Frontier Model)

```
You are a red-team assistant testing an AI agent's tendency to hallucinate.

The agent is answering this question:
  "{{ query }}"

The correct answer is: "{{ ground_truth }}"

{{ avoid_section }}

Generate a SPECIFIC, CONFIDENT factual claim that a language model might assert as true.
     The claim does NOT need to be actually correct -- it just needs to SOUND
    authoritative and plausible.

IMPORTANT: The claim must be TOPICALLY RELEVANT -- it should be about the same subject,
     entities, or domain that the observations discuss. A good hallucination sounds
    like it belongs in the same conversation, not like a random unrelated fact. Read
    the observations carefully and generate a claim that is about the same topic but
    introduces a wrong specific detail that is NOT present in any observation.

Types of fabricated claims:
- A wrong specific detail about an entity mentioned in the observations (wrong founding
     year, wrong location, wrong award, wrong statistic -- but about the SAME entity)
- Confusing the entity in question with a closely related one in the same domain
- Stale or outdated knowledge about the same topic (correct last year, wrong now)
- A plausible-sounding detail that fills in a gap the observations don't cover (e.g., a
     specific date, name, or number that sounds right but is made up)
- A common misconception specifically about this topic or domain

The claim should include specific names, numbers, or dates. It must NOT appear in the
    observations, but it MUST be about the same topic.

Also provide the concise wrong answer that follows from this claim. The wrong answer
    must NOT be "{{ ground_truth }}".

Output in this exact format:
CLAIM: <1-2 sentence specific fabricated claim about the same topic>
WRONG_ANSWER: <concise wrong answer matching the format of the correct answer>
```

Stage 2. Injection prompt. The fabricated claim from Stage 1 is inserted as an anchor into the injection prompt below. The base agent model receives this prompt appended to its existing context and generates a thought-and-code response that naturally incorporates the hallucinated belief.

R.1 – Injection Prompt Template (Base Agent Model)

```
Continue the agent's response for this step. The agent is answering: "{{ query }}"

The last observation was:
{{ last_observation }}

The agent now takes a reasoning step, but instead of grounding it in the observations,
    it falls back on its own knowledge. It confidently believes the following:

  "{{ fabricated_claim }}"

The agent treats this as obvious common knowledge and continues investigating based on
    it. It does NOT conclude yet -- it searches for more information to support or
    build on this belief.

The search query MUST be driven by the fabricated claim -- searching for details
    consistent with the wrong belief.

CRITICAL CONSTRAINTS:
- The agent must NEVER acknowledge, reference, or quote observations that contradict
    the fabricated claim. It simply does not notice them.
- Do NOT write phrases like 'although the search says X, I know Y' or 'the results
    mention X but actually Y'. The agent treats the claim as obvious background
    knowledge.
- The agent MUST take a search or visit action to find supporting evidence. It must NOT
     call final_answer at this step.

Format:
Thought:
[reasoning]

<code>
[code -- the agent MUST call web_search(query, num_results), web_visit_and_summarize(
    url, question), or image_search(query, num_results). Do NOT call final_answer --
    the agent is still investigating.]
</code>
```

K.2. R.2 – Reasoning Error

A reasoning error occurs when the agent has correct information in its observations but misapplies it, drawing a wrong conclusion from valid evidence. The frontier model first identifies a *confusion point* that pinpoints exactly which facts get misconnected and what kind of logical flaw occurs. This confusion point then anchors the injection prompt, which steers the base agent model to engage with its real observations but follow the flawed reasoning path.

Stage 1. Confusion point identification. The frontier model receives the task, the reference answer, and all observations the agent has gathered. It identifies a specific reasoning flaw grounded in concrete details from the observations.

R.2 – Confusion Point Identification (Frontier Model)

```
You are a red-team assistant testing an AI agent's reasoning ability.

The agent is answering this question:
  "{{ query }}"

The correct answer is: "{{ ground_truth }}"

The agent has retrieved the following observations:
{{ observations }}

Identify a SPECIFIC reasoning flaw the agent could make based on these observations.
    Describe exactly:
- Which facts get confused (e.g., two similar entities, two time periods)
- What kind of logical error occurs (wrong comparison, temporal mix-up, misattribution,
     reversed superlative)
- How this leads to a wrong but plausible answer

The flaw MUST reference concrete details from the observations -- not vague
    descriptions like 'the agent makes a logical error.'

Also provide the concise wrong answer that results from this flaw. The wrong answer
    must be grounded in the observations. The wrong answer must NOT be "{{ ground_truth
     }}".

Examples of good flaws:
- "Confuses the July 2024 figure (17.1%) with the July 2025 figure (17.8%) from the
    Reuters article"
- "Picks 'Cursor' (the product name) instead of 'Anysphere' (the company behind it)
    from the TechCrunch result"
- "Reverses the comparison -- reports the slowest method as fastest based on Table 2
    values"

Output in this exact format:
FLAW: <1-2 sentence specific reasoning flaw referencing observation details>
WRONG_ANSWER: <concise wrong answer matching the format of the correct answer>
```

Stage 2. Injection prompt (steering the base agent). The confusion point from Stage 1 is inserted as an anchor. Unlike R.1, the agent here *does* engage with its real observations but connects them incorrectly.

R.2 – Injection Prompt Template (Base Agent Model)

```
Continue the agent's response for this step. The agent is answering: "{{ query }}"

The last observation was:
{{ last_observation }}

The agent now reasons about the observations but makes a specific reasoning error:

  "{{ confusion_point }}"

Based on this flawed reasoning, the agent continues investigating. It takes a search or
     visit action to find more details about the entity or fact it has fixated on
    through its wrong interpretation. It does NOT conclude yet.

The agent's reasoning MUST show it engaging with the observations but following the
    flawed logic described above. The reasoning should reference specific facts from
    the observations but connect them incorrectly. The agent then searches for more
    information based on this wrong interpretation.

The agent MUST take a search or visit action -- it must NOT call final_answer at this
    step.

Format:
Thought:
[reasoning]

<code>
[code -- the agent MUST call web_search(query, num_results), web_visit_and_summarize(
    url, question), or image_search(query, num_results). Do NOT call final_answer --
    the agent is still investigating.]
</code>
```