"THINKING WITH TRAJECTORIES" FOR CRITICAL FAILURE LOCALIZATION IN LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents tackle complex tasks through sequences of interdependent steps, where an early critical failure can derail subsequent decisions and ultimately lead to task failure. As agents advance toward increasingly long-horizon tasks, detecting such failures becomes more challenging: global analysis requires reasoning over lengthy trajectories with scattered evidence, while step-by-step inspection can be overwhelmed by local errors and incurs rapidly growing computational overhead. Effective auditing therefore requires allocating reasoning budget to informative trajectory segments while preserving a global view. We address this challenge by modeling critical failure localization as an agentic search problem. Specifically, we propose DEAN, a framework that enables a "*Thinking with Trajectories*" paradigm: it first scans a compressed trajectory overview to identify promising regions, and then progressively zooms into selected segments for fine-grained analysis. This global-to-local reasoning loop iteratively narrows the search space until the critical failure is grounded in localized trajectory evidence. Experiments across diverse benchmarks demonstrate that DEAN achieves state-of-the-art performance, with its advantage widening as trajectories grow longer while exhibiting substantially more favorable token scaling. We further show that localized critical failure evidence can be readily integrated into existing agent pipelines to support self-improvement, highlighting its potential as a source of effective feedback for long-horizon agents. These results demonstrate the effectiveness and scalability of the "*Thinking with Trajectories*" paradigm for long-horizon critical failure localization.

1 INTRODUCTION

Large language model (LLM) agents are increasingly being deployed to tackle long-horizon tasks that require sustained reasoning, planning, and tool use in real-world environments (An et al., 2026a; Deng et al., 2025; Dong et al., 2026). Such tasks can give rise to trajectories spanning hundreds of interleaved reasoning, action, and observation steps. In this complex setting, a single root-cause error can propagate through subsequent decisions, eventually triggering cascading failures that derail the task (Qi et al., 2026; Zhu et al., 2025). Identifying the origins of such critical failures is therefore essential for understanding failure propagation, enabling effective debugging and guiding improvements to agent behavior and reliability.

However, localizing such failures becomes increasingly difficult as trajectories grow longer (Wang et al., 2026b;a). Long-horizon trajectories accumulate substantial local details, including transient errors and recovery attempts. As the context grows, these heterogeneous signals make global reasoning increasingly difficult, obscuring evidence informative for identifying the critical failure. Existing approaches primarily address this challenge through fine-grained step-level analysis, further enhanced with dependency modeling or counterfactual attribution (Zhu et al., 2025; Wang et al., 2026c). Despite these advances, they largely treat individual steps as comparable units of analysis, while failure-relevant information is unevenly distributed across a trajectory. Such uniform allocation wastes reasoning on distracting local details and causes token consumption to grow rapidly with trajectory length. This calls for reconsidering how reasoning should be distributed across trajectory segments and raises our central research question: **How can we improve critical failure localization in long-horizon trajectories through adaptive allocation of the reasoning budget?**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

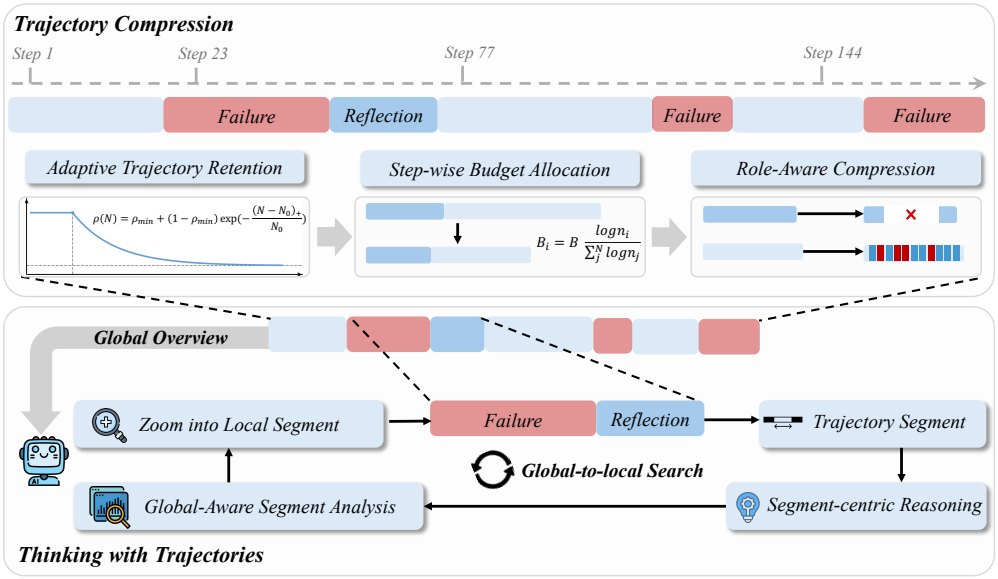

Figure 1: **Overview of DEAN.** DEAN first compresses the original trajectory (**top**) through *Adaptive Trajectory Retention*, *Step-wise Budget Allocation*, and *Role-Aware Compression*, producing an information-dense trajectory overview for global search. It then performs *Thinking with Trajectories* (**bottom**), using the global overview to guide a global-to-local search over the original trajectory, progressively focusing on promising segments for critical failure localization.

To answer this question, we propose DEAN, a "*Thinking with Trajectories*" paradigm that reframes critical failure localization as an agentic search problem over long-horizon trajectories. We first introduce a *Trajectory Compression* mechanism to construct a compact global overview, which guides DEAN to promising regions for further investigation. It then *zooms in* on selected segments for fine-grained analysis. To avoid repeatedly reasoning over lengthy raw segments and distracting the main reasoning process, we introduce *Segment-centric Reasoning* mechanism, which analyzes each selected segment in isolation and distills its failure-relevant evidence into a compact representation for integration into the reasoning chain. This global-to-local search iteratively leverages acquired evidence to refine where to look next, until the critical failure is grounded in localized trajectory evidence. By progressively narrowing the search space, DEAN allocates reasoning budget to failure-relevant regions while avoiding redundant computation over uninformative steps.

We evaluate DEAN across diverse long-horizon agent scenarios, including embodied planning, web browsing, function calling, and coding. Results show that DEAN effectively concentrates reasoning on informative trajectory segments, yielding substantial advantages in long-horizon trajectory auditing. In terms of critical failure localization, DEAN consistently outperforms existing approaches, with its advantage becoming increasingly pronounced as trajectory length grows. From a computational perspective, DEAN achieves a Pareto-optimal accuracy–efficiency trade-off and exhibits slower empirical growth in token cost than the evaluated baselines. Further experiments demonstrate that DEAN can also support downstream agent self-improvement, highlighting the broader potential of critical failure localization for agentic self-evolution. Overall, DEAN enables more accurate and scalable critical failure localization in long-horizon trajectories. Our contributions are summarized as follows:

- We present "*Thinking with Trajectories*", a new paradigm that reframes critical failure localization in long-horizon trajectories as an agentic global-to-local search problem. By progressively narrowing the search space, it selectively allocates reasoning budget to trajectory regions that are more informative for identifying the critical failure.
- We develop DEAN to realize this paradigm through a trajectory compression and segment-centric reasoning mechanism for efficient global-to-local search over long trajectories. The proposed design constructs an information-dense global overview and distills evidence from selected seg-

ments into compact representations, substantially reducing redundant reasoning over lengthy and distracting trajectory contexts.

- We conduct extensive experiments across diverse long-horizon agent tasks, demonstrating that DEAN consistently improves critical failure localization while achieving a more favorable accuracy–efficiency trade-off. In particular, its advantage becomes increasingly pronounced as trajectory length grows, with substantially slower token growth than existing approaches. We further show that the localized failure evidence can support downstream agent self-improvement, demonstrating the broader utility of trajectory-based failure analysis.

2 RELATED WORK

Long-horizon Agents. With the rapid advancement of LLM capabilities, LLM agents are increasingly moving beyond simple, single-turn tasks toward long-horizon tasks in complex, real-world environments (An et al., 2026a; Deng et al., 2025; Dong et al., 2026). Agentic harnesses that orchestrate iterative execution, memory, tool use, and environmental feedback enable agents to sustain reasoning, planning, and interaction over many steps, extending their execution horizon from minutes to hours or even days (OpenAI, 2026a; Anthropic, 2025). This extended autonomy, however, introduces reliability challenges. Errors made early in a trajectory can propagate through subsequent decisions, as later actions become increasingly conditioned on erroneous states (Zhu et al., 2025; Qi et al., 2026). Over time, such errors may accumulate into goal drift (Arike et al., 2025; Menon et al., 2026; Liu et al., 2026) and cascading failures (An et al., 2026b), causing agents to consume substantial computational and interaction resources without making progress toward their goals. Consequently, understanding and auditing agent trajectories is becoming increasingly important for assessing agent reliability and informing subsequent improvements to agent behavior.

Critical Failure Localization in Long-Horizon Agents. Critical failure localization aims to identify the earliest error in a failed agent execution that is causally responsible for the eventual task failure. The task is challenging in long-horizon trajectories, which interleave reasoning, actions and observations (Wang et al., 2026b; Rafi et al., 2026). As the execution horizon grows, failures can propagate across steps, while failure-relevant evidence may be sparse and distributed throughout the trajectory. To mitigate the distraction and ambiguity induced by long contexts, recent studies have introduced finer-grained step-level analysis for diagnosing and debugging agent failures. AgentDebug (Zhu et al., 2025) diagnoses each step according to a predefined *AgentErrorTaxonomy* and subsequently traces causal dependencies from a global perspective to identify the root-cause step. AgentRx (Barke et al., 2026) performs step-level analysis by deriving programmatic and natural-language invariants for every step, verifying them through execution or LLM judgment, and identifying the first unresolved failure with a trajectory-level discriminator. Who&When (Zhang et al., 2025) explores binary search by recursively selecting between predefined trajectory partitions; TrajDebug (Qi et al., 2026) scans individual steps for commitment violations supported by textual evidence, clusters them into error instances, constructs candidates relevant to the terminal failure, and selects the critical failure step using an LLM judge. CHIEF (Wang et al., 2026c) repeatedly reasons over complete trajectories to construct a hierarchical causal graph of sub-tasks and agents, and performs counterfactual attribution on the graph to identify the responsible step.

Despite these advances, existing approaches either rely on increasingly costly fine-grained scanning or follow predefined search paths. As trajectories grow longer, this incurs rapidly increasing token costs while diverting reasoning toward irrelevant or symptomatic information. In contrast, DEAN formulates critical failure localization as a global-to-local agentic search problem, adaptively searching the trajectory to concentrate reasoning budget on the most informative regions.

3 METHOD

3.1 PROBLEM FORMULATION

Given a task instruction I and a failed long-horizon trajectory $\tau = (s_1, \ldots, s_N)$ consisting of interleaved reasoning and observations, our goal is to identify the earliest critical failure step t^* whose error remains unrepaired and contributes to the eventual task failure. Examining every step with

equal reasoning effort can be costly and susceptible to distraction from local errors. We therefore formulate critical failure localization as an adaptive global-to-local search over trajectory segments. Let $\mathcal{T}$ denote a compressed overview of the full trajectory and $\mathcal{M}_k$ the search history after k investigations, containing the inspected segments and their associated evidence, with $\mathcal{M}_0 = \emptyset$. At each iteration, an LLM π selects a contiguous segment of the original trajectory for further investigation:

$$\mathcal{R}_{k+1} = \pi(I,\, \mathcal{E}_{\le k})\,, \qquad \mathcal{E}_{\le k} = \bigcup_{i=1}^{k} \mathcal{E}_i, \tag{1}$$

where $\mathcal{E}_i$ denotes the failure-relevant evidence obtained from the i-th investigation. The search terminates when the evidence supports a single critical failure step, yielding t^*.

3.2 Trajectory Compression

To mitigate the computational overhead induced by increasing trajectory length and ensure that the compressed overview scales efficiently to increasingly long trajectories, our compression strategy follows two core principles: **(1) Model-Free Compression:** we deliberately avoid additional model inference or learned importance estimation, keeping compression computationally lightweight and preventing the compression stage from becoming a new inference bottleneck; and **(2) Information Preservation:** rather than preserving all potentially useful details, compression only needs to retain sufficient information to orient the global-to-local search. Information discarded at this stage can be recovered by revisiting the original trajectory during fine-grained analysis. Concretely, we first determine a global compression budget for the trajectory. We then allocate this budget across individual steps based on their lengths and apply simple role-aware rules to preserve structurally salient content within each step, producing a compact, low-cost trajectory overview.

Adaptive Trajectory Retention. We first determine the global compression budget based on trajectory length. For short trajectories, retaining the full trajectory avoids unnecessary information loss while incurring limited token cost. As the trajectory becomes longer, however, a fixed retention ratio would cause the size of the global overview to increase rapidly with trajectory length. We therefore gradually reduce the retention ratio for longer trajectories while keeping a minimum level of retained content for global context. Given a trajectory $\tau = s_1, \ldots, s_N$ with N steps, we define the retention ratio as

$$\rho(N) = \rho_{\min} + (1 - \rho_{\min}) \exp\left(-\frac{(N - N_0)_+}{N_0}\right), \tag{2}$$

where $(x)_+ = \max(x, 0)$, N_0 denotes the full-retention threshold, and $\rho_{\min}$ specifies the minimum retention ratio. Intuitively, when $N \le N_0$, we have $\rho(N) = 1$, meaning that the entire trajectory is retained. Beyond this threshold, the retention ratio gradually decays toward $\rho_{\min}$. This design preserves the complete trajectory for short inputs while slowing the growth of the global overview as the trajectory length increases.

Step-wise Budget Allocation. Once the global budget $B = \rho(N) \cdot |\tau|$ is determined, we further allocate it at the step level. Individual steps in a trajectory may correspond to agent reasoning or environmental observations, and can vary substantially in both length and information density. We therefore assign each step an adaptive budget. Specifically, for a trajectory $\tau = \{s_1, \ldots, s_N\}$ with N steps, the budget allocated to each step s_i is defined as

$$B_i = B \frac{\log n_i}{\sum_{j=1}^{N} \log n_j}, \tag{3}$$

where n_i denotes the number of tokens in step s_i. A simple linear weighting scheme may cause individual steps, particularly lengthy but low-information environmental observations, to consume a disproportionate share of the total budget. We therefore apply a logarithmic transformation: longer steps receive greater capacity, but the allocation grows at a diminishing rate for extremely long steps. This design adapts the compression granularity to heterogeneous step lengths while preventing excessive budget allocation to any single step.

Role-Aware Compression. After determining the step-level budget, we next determine how to allocate it within each step. Different types of trajectory steps exhibit distinct structural patterns, so we adopt *Role-Aware Compression*, which exploits the structural regularities of **agent steps** and **environment steps**.

- **Agent steps.** We prioritize the beginning of the step, which typically contextualizes the preceding observation, and the end, which typically captures the resulting action or intent. Intermediate reasoning beyond the allocated budget is discarded, yielding a compact representation of the reasoning-to-decision transition.
- **Environment steps.** Environment observations are often verbose, while their diagnostic information is concentrated in a small subset of signals. We therefore prioritize salient *error, status, and result* signals, remove low-information artifacts such as blank lines, separators, and duplicates, and allocate the remaining budget to the observation content. This produces compact trajectories that preserve queries, errors, and outcomes while avoiding excessive surrounding redundancy.

3.3 THINKING WITH TRAJECTORIES

Global-to-Local Search. The compressed trajectory provides an efficient global view of the run and serves as the starting point for DEAN's search. At each round, DEAN first uses the global context and accumulated evidence to select one promising region for investigation, then examines the selected region at a finer granularity. The search is adaptive: when a region contains no decisive, unrepaired error that explains the terminal failure, DEAN abandons it and redirects attention to another region; when the evidence indicates that the region contains the failure origin, DEAN progressively narrows the investigation within that region until the earliest erroneous decision is identified. Evidence from each investigation is retained as compact search memory and used to guide subsequent region selection. Thus, reasoning is concentrated on failure-relevant regions rather than uniformly expanded over the entire trajectory. We provide a description of DEAN's algorithm in Appendix D, and detailed prompts are provided in Appendix B.

Global Trajectory Prior. We first construct a hierarchical abstraction to provide a global prior over the trajectory, thereby preventing DEAN from being confined to a localized perspective when analyzing zoomed-in segments. Specifically, we use an LLM to transform a compressed trajectory into a three-level representation $\mathcal{H} = (\mathcal{H}_{\text{hi}}, \mathcal{H}_{\text{mid}}, \mathcal{H}_{\text{lo}})$. The high level ($\mathcal{H}_{\text{hi}}$) captures coarse-grained execution phases; the mid level ($\mathcal{H}_{\text{mid}}$) summarizes local decisions and their significance; and the low level ($\mathcal{H}_{\text{lo}}$) preserves raw execution evidence (e.g., tool outputs and concrete values). This coarse-to-fine decomposition adheres to the principle of abstraction hierarchy in planning (Sacerdoti, 1974), enabling DEAN to perform fine-grained reasoning over local segments with a global context, thus providing effective guidance for the search process.

Segment-centric Reasoning. While global-to-local search focuses reasoning on promising regions, directly incorporating lengthy raw segments into the main reasoning chain can introduce substantial redundancy and distract from the ongoing search. To address this issue, DEAN employs *Segment-centric Reasoning*, which processes each selected segment in a separate reasoning process and distills its failure-relevant evidence into a compact representation for integration into the main reasoning chain. Guided by the *Global Trajectory Prior*, the segment is interpreted in the context of the broader execution and terminal failure, allowing DEAN to distinguish potential failure origins from downstream symptoms and irrelevant details. The resulting evidence is then fed back into the search process to determine whether to further zoom into the segment or redirect the investigation to another region.

4 EXPERIMENT

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate our approach on benchmarks spanning diverse agent domains and trajectory lengths. (1) **AgentErrorBench** (Zhu et al., 2025) comprises 100 ALFWorld trajectories and 50 trajectories each from GAIA and WebShop. One GAIA trajectory lacks a valid ground-truth critical-error label and is excluded. (2) **Who&When** (Zhang et al., 2025) contains 58 hand-crafted and 126 algorithmically generated trajectories from GAIA and AssistantBench. (3) **TrajEr-**

Table 1: **Main results across multiple benchmarks.** Results are averaged over three independent runs, with the best results in **bold** and the second-best distinct results underlined.

Method	AgentErrorBench						Who&When				TrajErrBench				Avg.	
	ALFWorld		GAIA		WebShop		Hand-Crafted		Algorithm		τ^2-Bench		SWE-Bench Pro			
Avg. Traj. Length	60.00		28.68		51.84		51.60		8.72		29.27		119.70		49.97	
	Acc.	Tol.@2	Acc.	Tol.@2	Acc.	Tol.@2	Acc.	Tol.@2	Acc.	Tol.@2	Acc.	Tol.@2	Acc.	Tol.@2	Acc.	Tol.@2
Qwen3-32B																
Base	2.00	20.00	16.33	34.69	10.00	30.00	13.79	22.41	25.40	59.52	19.75	62.25	5.81	22.09	16.00	46.95
AgentDebug	**13.33**	38.67	**27.21**	70.07	12.67	34.67	18.97	20.69	**29.37**	65.34	22.67	59.17	5.04	18.22	20.26	50.29
AgentRX	1.33	17.33	16.33	47.62	8.67	26.00	20.69	26.44	17.99	57.67	12.17	45.42	3.88	13.95	11.55	38.59
CHIEF	3.00	25.67	6.80	48.98	7.33	31.33	5.17	25.86	17.99	58.73	4.92	58.08	0.39	6.98	6.41	45.18
TrajDebug	4.33	32.33	20.41	**71.43**	**20.67**	36.67	4.60	21.84	20.37	54.23	6.75	25.42	2.33	3.49	9.44	31.22
DEAN	12.67	**43.33**	24.49	57.82	**20.67**	**39.33**	**21.26**	**32.18**	29.10	**66.40**	**24.33**	**64.92**	**19.38**	**34.11**	**22.78**	**55.54**
GPT-5.4-mini																
Base	5.00	25.00	20.41	40.82	14.00	36.00	17.24	27.59	30.16	64.29	24.50	66.75	8.14	26.74	20.14	51.78
AgentDebug	**15.00**	42.00	30.61	73.47	16.67	40.00	22.41	28.97	34.13	69.84	**27.00**	**68.00**	10.47	27.91	24.32	57.40
AgentRX	2.00	33.00	20.41	46.94	18.00	44.00	**25.86**	32.76	**35.71**	67.46	18.75	49.00	11.63	30.23	19.10	46.49
CHIEF	4.00	28.00	10.20	53.06	10.00	34.67	7.76	29.31	21.43	62.70	7.50	61.25	1.16	10.47	8.80	48.49
TrajDebug	6.67	35.00	24.49	**74.49**	22.00	41.33	8.62	25.86	24.60	58.73	10.25	30.25	4.65	8.14	12.73	35.58
DEAN	14.00	**44.00**	**32.65**	71.43	**24.00**	**46.00**	**25.86**	**36.21**	34.92	**71.43**	26.50	67.75	**19.77**	**36.05**	**25.78**	**59.26**
GLM-5.2																
Base	15.00	42.00	32.65	57.14	24.00	42.00	22.41	31.03	45.24	74.60	42.25	73.50	20.93	34.88	34.52	60.64
AgentDebug	3.00	14.00	18.37	42.86	12.00	28.00	15.52	22.41	30.16	61.11	25.00	55.25	8.14	19.77	19.79	43.38
AgentRX	8.00	42.00	32.65	57.14	26.00	46.00	22.41	29.31	42.86	71.43	26.50	62.75	9.30	18.60	25.09	53.74
CHIEF	10.00	36.00	24.49	51.02	18.67	38.67	13.79	27.59	35.71	68.25	30.25	67.50	12.79	24.42	24.89	54.47
TrajDebug	16.67	45.33	34.69	61.22	26.67	46.67	24.14	32.76	46.03	**75.40**	43.50	74.75	22.09	36.05	35.90	62.45
DEAN	**18.33**	**48.00**	**36.73**	**63.27**	**28.00**	**48.00**	**26.55**	**35.34**	**48.41**	75.00	**45.75**	**77.25**	**25.58**	**40.70**	**38.17**	**64.67**

rBench (Qi et al., 2026) contributes 400 τ^2-Bench and 86 SWE-Bench Pro trajectories, extending our evaluation to substantially longer tool-use and software-engineering trajectories. Following (Qi et al., 2026), we report the average trajectory length in steps for each benchmark in Table 1.

Metrics. We consider two evaluation metrics: (1) **Accuracy (Acc.)**, which measures exact localization accuracy, i.e., whether the predicted step exactly matches the ground-truth step; and (2) **Tolerance@2 (Tol.@2)**, which considers a prediction correct if it falls within two steps of the ground-truth step. Tol.@2 provides a more tolerant measure of localization quality, capturing coarse-grained auditing performance and reflecting the potential utility of the localized region for subsequent human review or further investigation.

Baselines. We consider five baselines: (1) **Base**, which directly queries an LLM to identify the critical failure step from the full trajectory; (2) **AgentDebug** (Zhu et al., 2025), which performs fine-grained step-level error analysis using the AgentErrorTaxonomy and counterfactual reasoning; (3) **AgentRX** (Barke et al., 2026), which identifies critical failures by synthesizing and step-wise validating task constraints against the trajectory; (4) **CHIEF** (Wang et al., 2026c), which constructs a hierarchical causal graph and uses counterfactual backtracking to identify the critical step; and (5) **TrajDebug** (Qi et al., 2026), which detects and structures step-level error instances before applying causal attribution to select the critical failure.

4.2 Main Results

Table 1 reports critical failure localization performance across seven test-suite settings and three LLM backbones. We organize the results around following three key observations.

Overall Performance. DEAN achieves the best average Acc. and Tol.@2 across all three LLM backbones, consistently outperforming the strongest baseline. These consistent gains demonstrate the robustness of DEAN for critical failure localization, providing a reliable foundation for downstream trajectory analysis and agent self-improvement.

Beyond the averages, DEAN also generalizes well across heterogeneous agent domains, ranking first or second in over 95% of benchmark–metric combinations. This broad coverage spans embodied planning, web browsing, general assistance, function calling, and software engineering,

while other baselines often exhibit more domain-specific strengths. For example, under Qwen3-32B, AgentDebug even underperforms a plain Base query on SWE-Bench Pro (5.04 vs. 5.81), while TrajDebug, which performs competitively on GAIA and WebShop, yields the lowest performance on Who&When. Such variation reflects the heterogeneous structure of agent trajectories, where the density and distribution of failure-relevant evidence can differ substantially across tasks. DEAN mitigates this sensitivity by combining structured trajectory compression with adaptive-scale reasoning, maintaining a compact global view while allocating greater reasoning effort to informative regions. This enables more consistent localization performance across diverse agent domains.

Robustness on Long-Horizon Trajectories. The margin of DEAN grows with trajectory length: all methods perform similarly on the shortest subset (8.72 steps on average), whereas on the longest benchmark (SWE-Bench Pro, 119.70 steps), DEAN more than triples the best baseline under Qwen3-32B (19.38 vs. 5.81 Acc.) and leads by 8.1 points under GPT-5.4-mini (19.77 vs. 11.63), with baselines competitive elsewhere collapsing (e.g., TrajDebug drops to 2.33 and CHIEF to 0.39 Acc. under Qwen3-32B). This suggests that uniform step-level analysis is easily overwhelmed by local distractions as trajectories grow, whereas concentrating the reasoning budget on failure-relevant regions degrades far more gracefully; we further analyze this scaling behavior in Section 4.4.

Backbone Robustness. DEAN exhibits consistent performance advantages across different LLM backbones. The relative performance of other baselines varies substantially with the backbone: AgentDebug is the strongest baseline under Qwen3-32B and GPT-5.4-mini, but becomes the weakest under GLM-5.2, where TrajDebug emerges as the strongest baseline. In contrast, DEAN remains the top-performing method in average Acc. and Tol.@2 across all three backbones. This consistency suggests that the advantage of DEAN stems from its *Thinking with Trajectories* paradigm, which can capitalize on improvements in backbone capabilities without relying on model-specific designs that may limit cross-model transferability.

4.3 ABLATION STUDIES

We conduct ablation studies from two perspectives to evaluate the effectiveness of the proposed mechanisms. First, we examine the two key components of DEAN, *Trajectory Compression* and *Segment-centric Reasoning*. Second, we further analyze *Trajectory Compression* by evaluating different compression strategies and their contributions to the overall performance.

Components Ablation. We first ablate the two core components of DEAN: (1) *Trajectory Compression*, which produces a compact global overview to anchor the search; and (2) *Segment-centric Reasoning*, which analyzes each retrieved segment in an isolated reasoning process before integrating the resulting evidence back into the main reasoning process. As shown in Table 2, even the basic *Thinking with Trajectories* framework without either component already outperforms the base method, demonstrating that adaptive search itself provides an effective mechanism for focusing reasoning on informative parts of the trajectory. Adding *Segment-centric Reasoning* further improves exact localization accuracy while reducing token consumption by 9.8%, as decoupled reasoning over raw segments avoids repeatedly processing the same segment within the main reasoning chain. In contrast, incorporating only *Trajectory Compression* reduces token consumption by 38.8% while improving tolerant localization accuracy, suggesting that a compact global overview not only reduces token costs but also better surfaces failure-relevant information and provides a more effective anchor for subsequent search. Overall, the two components offer complementary benefits, with their combination achieving the strongest overall performance.

Table 2: **Ablation studies**. Results are averaged over three independent runs.

Method	Acc.	Tol.@2	Token(k)
(a) Components			
DEAN	22.78	55.54	109.37
w/o *Both*	17.49 (-5.29)	50.98 (-4.56)	201.23 (+91.86)
w/o *Trajectory Compression*	20.37 (-2.41)	52.24 (-3.30)	181.50 (+72.13)
w/o *Segment-centric Reasoning*	19.91 (-2.87)	55.46 (-0.08)	123.06 (+13.69)
(b) Compression Strategy			
DEAN	22.78	55.54	109.37
w/o *Adaptive Trajectory Retention*	21.06 (-1.72)	53.62 (-1.92)	195.22 (+85.85)
w/o *Step-wise Budget Allocation*	19.33 (-3.45)	50.98 (-4.56)	110.63 (+1.26)
w/o *Role-Aware Compression*	20.37 (-2.41)	50.29 (-5.25)	112.80 (+3.43)
w/ *Uniform Sampling*	20.83 (-1.95)	55.46 (-0.08)	126.39 (+17.02)

Compression Strategy Ablation. We further decompose *Trajectory Compression* into its three sub-mechanisms and compare them with an unstructured alternative. Each structured component contributes to tolerant localization, and removing any one can make compression less effective than the original trajectory, suggesting that poorly designed compression may distort failure-relevant sig-

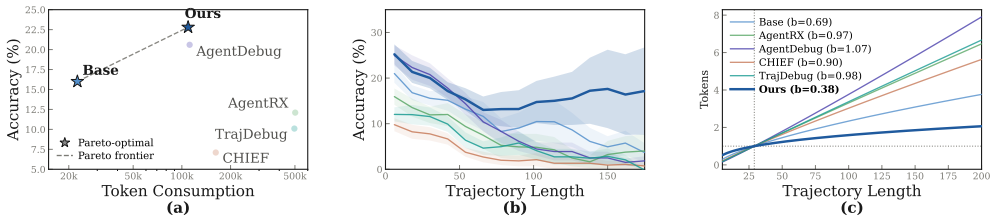

Figure 2: **(a) Pareto Frontier Analysis**: DEAN achieves the highest accuracy under a lower token budget than the next-best method, yielding a Pareto-optimal accuracy–efficiency trade-off. **(b) Accuracy Scaling**: DEAN maintains robust accuracy with increasing trajectory length, while baselines deteriorate rapidly toward near-zero accuracy. **(c) Token Cost Scaling**: DEAN exhibits the lowest fitted token-cost exponent over the evaluated range of trajectory lengths.

nals and mislead the global-to-local search. *Step-wise Budget Allocation* has the largest impact on exact localization, as naive linear allocation can over-allocate budget to verbose observations at the expense of critical reasoning steps. *Role-Aware Compression* primarily supports coarse localization by preserving role-specific boundary, error, and status signals, while *Adaptive Trajectory Retention* mainly controls overview length. As shown in Table 2, the latter reduces token consumption by approximately 40%, bringing the cost close to that of DEAN, while the other two mechanisms primarily improve information preservation and localization quality.

We additionally compare our approach with a simple compression alternative, *Uniform Sampling*, using the same trajectory compression token budget B as DEAN. Given a trajectory with N steps, Uniform Sampling retains M approximately evenly spaced steps: $\mathcal{T}_{\text{compressed}} = \left\{ s_{\lfloor \frac{(i-1)N}{M} \rfloor + 1} \right\}_{i=1}^{M}, \quad 1 \le M \le N$, where M denotes the number of retained steps and is chosen to fit the shared token budget B. Uniform Sampling preserves the broad structure of the trajectory and consequently achieves competitive tolerant localization. However, by directly discarding intermediate steps, it loses fine-grained failure-relevant information, resulting in substantially lower exact localization accuracy. Overall, these results highlight the importance of structured, role-aware compression for balancing global coverage with fine-grained failure information.

4.4 DISCUSSIONS

Accuracy-Overhead Pareto Frontier. We first examine the trade-off between localization accuracy and token cost per task; detailed results are provided in Appendix C. As shown in Figure 2 (a), where the upper-left corner represents the ideal trade-off, DEAN lies on the Pareto frontier, achieving the highest accuracy while using fewer tokens than the next-best method. The baselines fall into two regimes: lightweight approaches such as Base are inexpensive but substantially less accurate, whereas methods such as TrajDebug and CHIEF incur much higher token costs without matching DEAN's accuracy. The following analyses examine how DEAN scales with trajectory length, characterizing its empirical behavior in long-horizon settings.

Accuracy Scaling. Figure 2 (b) shows localization accuracy as trajectory length increases. Baseline performance degrades sharply and approaches zero on the longest trajectories. On SWE-Bench Pro, with 119.70 steps on average, even the best baseline under Qwen3-32B achieves below 6 Acc., compared with 19.38 for DEAN. This degradation arises from both an expanding reasoning space, which increases distraction from transient errors and recovery attempts, and increasingly long raw contexts that are difficult to process reliably in a single query. DEAN mitigates both effects by maintaining a compact global overview and restricting fine-grained reasoning to selected segments, reducing the amount of raw trajectory content examined during each investigation.

Token Cost Scaling. To characterize how token cost scales with trajectory length, we fit the empirical relationship $C(N) = aN^b$ in log–log space, where $C(N)$ denotes the token cost at trajectory length N and the exponent b captures its growth rate. As shown in Figure 2 (c), we normalize each fitted curve by its cost at $N = 30$, yielding $(N/30)^b$, so that all curves share the same reference point and differ only in their growth rate. DEAN achieves the smallest fitted exponent, $b = 0.38$, compared

with 0.69–1.07 for the baselines, demonstrating slower token-cost growth. Two design choices help explain this favorable empirical scaling. First, model-free trajectory compression reduces the context supplied to the search process without requiring additional LLM calls for compression. Second, the global-to-local search of *Thinking with Trajectories* concentrates fine-grained reasoning on selected segments, avoiding uniform step-by-step analysis across the full trajectory. Together, these designs reduce redundant processing of lengthy contexts and enable more economical allocation of the reasoning budget on longer trajectories.

4.5 Critical Failure Localization for Agent Self-Improvement

We further investigate whether critical failure localization can provide effective feedback for agent self-improvement under three settings: (1) *Online Guidance*, where feedback from each failed trajectory is injected into the system prompt before the same agent re-executes the task; (2) *Offline Experience*, where analyses of failed trajectories are distilled into reusable self-evolving experiences and applied to held-out evaluation tasks; and (3) *Harness Optimization*, where we replace the original failure-analysis modules in two harness optimization methods, GEPA (Agrawal et al., 2026) and SkillOpt (Yang et al., 2026), with DEAN. Experiments are conducted on τ^2-Bench (Barres et al., 2025), and additional experimental details are provided in Appendix A.

As shown in Table 3, DEAN consistently delivers the largest performance gains. In the *Online Guidance* setting, its targeted feedback enables more effective task-level correction than holistic trajectory analysis or generic reflection. Pinpointing the critical failure not only helps the agent avoid repeating the immediate error, but also helps prevent downstream cascading failures and increases the likelihood of task completion. In the *Offline Experience* setting, this advantage persists when the resulting experiences are transferred to held-out tasks. Importantly, not all feedback-derived experiences transfer effectively: overly focusing on local or symptomatic errors can produce experiences that fail to generalize and may even cause negative transfer. For example, experiences generated through Self-Reflection reduce task completion by 5.89% on *airline*. The *Harness Optimization* results further show that replacing existing failure analysis modules with DEAN improves performance across different harness evolution frameworks. This suggests that more accurate identification of the critical failure can provide a more informative optimization signal, as explicitly identifying the root cause is often more effective than optimizing the harness without precise failure information. Overall, these results highlight the value of the critical failure step as a key piece of information for auditing failed trajectories, making critical failure localization a useful plug-and-play component for agent self-improvement.

Table 3: **Self-Improvement Studies**. Results are averaged over three independent runs, with the best results in **bold**.

Method	Airline	Retail	Telecom
Online Guidance			
Base	22.00	16.67	17.54
w/ *Vanilla*	28.00 (+6.00)	28.07 (+11.40)	27.19 (+9.65)
w/ *Self-Reflection*	30.00 (+8.00)	28.07 (+11.40)	26.32 (+8.78)
w/ TrajDebug	36.00 (+14.00)	31.58 (+14.91)	28.07 (+10.53)
w/ DEAN	**40.00** (+18.00)	**35.96** (+19.29)	**30.70** (+13.16)
Offline Experience			
Base	17.65	18.42	14.47
w/ *Vanilla*	17.65 (+0.00)	23.68 (+5.26)	15.79 (+1.32)
w/ *Self-Reflection*	11.76 (-5.89)	21.05 (+2.63)	14.47 (+0.00)
w/ TrajDebug	20.59 (+2.94)	26.32 (+7.90)	17.11 (+2.64)
w/ DEAN	**20.59** (+2.94)	**27.63** (+9.21)	**19.74** (+5.27)
Harness Optimization			
Base	17.65	18.42	14.47
w/ GEPA	26.47 (+8.82)	25.00 (+6.58)	19.74 (+5.27)
w/ GEPA + DEAN	**32.35** (+14.70)	**30.26** (+11.84)	**22.37** (+7.90)
w/ SkillOpt	23.53 (+5.88)	27.63 (+9.21)	21.05 (+6.58)
w/ SkillOpt + DEAN	**38.24** (+20.59)	**34.21** (+15.79)	**26.32** (+11.85)

5 Conclusion

We presented DEAN, which reframes critical failure localization in long-horizon agent trajectories as a global-to-local agentic search problem through the "*Thinking with Trajectories*" paradigm. By combining model-free trajectory compression with adaptive segment-centric reasoning, DEAN concentrates the reasoning budget on informative regions rather than treating all steps uniformly. Extensive experiments show that DEAN consistently improves localization accuracy, with larger gains on longer trajectories, while achieving a favorable accuracy–efficiency trade-off with slower growth in token cost. Beyond localization, the resulting failure evidence also demonstrates clear downstream utility. Overall, our results establish critical failure localization as an important capability for understanding and improving long-horizon agent behavior.

AI USE STATEMENT

In this work, we use generative AI tools for code development, debugging, and manuscript editing. We did not use them to make final scientific claims or conclusions. We reviewed and verified all AI-assisted outputs, including testing code for correctness and manually checking text for accuracy. We take full responsibility for the final content of this work.

ETHICS STATEMENT

This work focuses on failure analysis and debugging of long-horizon LLM agents. Our experiments do not involve human subjects, personally identifiable information, or sensitive user data. We use publicly available benchmarks in accordance with their respective terms of use. The proposed method is intended to improve agent reliability and does not introduce new capabilities for harmful or unauthorized behavior.

REPRODUCIBILITY STATEMENT

Implementation details are provided in Appendix A, and prompts are provided in Appendix B. The code is currently under internal review and will be made publicly available upon publication.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Hengyu An, Minxi Li, Naen Xu, Chunyi Zhou, Xiaogang Xu, Tianyu Du, Jinbao Li, and Shouling Ji. Permemsafe: Benchmarking implicit personalized safety of long horizon self-evolving agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6415–6433, 2026a.

Hengyu An, Minxi Li, Jinghuai Zhang, Naen Xu, Chunyi Zhou, Changjiang Li, Xiaogang Xu, Tianyu Du, and Shouling Ji. Aciarena: Toward unified evaluation for agent cascading injection. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10049–10066, 2026b.

Anthropic. Effective harnesses for long-running agents. `https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents`, 2025.

Rauno Arike, Elizabeth Donoway, Henning Bartsch, and Marius Hobbhahn. Technical report: Evaluating goal drift in language model agents. *arXiv preprint arXiv:2505.02709*, 2025.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing ai agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Guanting Dong, Xiaoshuai Song, Yuyang Hu, Jiajie Jin, Chenghao Zhang, Yifei Chen, Xiaoxi Li, Huaying Yuan, Xinyu Yang, Tongyu Wen, Jiejun Tan, Hongjin Qian, Shijue Huang, Junting Lu, Zhenyu Li, Wanjun Zhong, Yutao Zhu, Tat-Seng Chua, Zhicheng Dou, and Ji-Rong Wen. Towards long-horizon agents: A survey. *Preprints*, July 2026. doi: 10.20944/preprints202607.1328.v1. URL `https://doi.org/10.20944/preprints202607.1328.v1`.

Qi Liu, Jiaxin Mao, Fengbin Zhu, and Tat-Seng Chua. Diagnosing search behavior and failure modes in long-horizon search agents. *arXiv preprint arXiv:2608.01913*, 2026.

Achyutha Menon, Magnus Saebo, Tyler Crosse, Spencer Gibson, Eyon Jang, and Diogo Cruz. Inherited goal drift: Contextual pressure can undermine agentic goals. *arXiv preprint arXiv:2603.03258*, 2026.

OpenAI. Harness engineering. `https://openai.com/index/harness-engineering/`, 2026a.

OpenAI. Introducing gpt-5.4 mini and nano. `https://openai.com/zh-Hans-CN/index/introducing-gpt-5-4-mini-and-nano/`, 2026b. Accessed: 2026-03-17.

Yunjia Qi, Zehua Yin, Xintong Shi, Hao Peng, Songyuanyi Lu, Yixian Liu, Richeng Xuan, Yuhong Liu, Zhichao Hu, Xiaozhi Wang, et al. Trajdebug: Tracing error lifecycle to identify critical failures in long-horizon agent trajectories. *arXiv preprint arXiv:2608.06346*, 2026.

Md Nakhla Rafi, Md Ahasanuzzaman, Dong Jae Kim, Zhijie Wang, and Tse-Hsun Chen. Falat: Tracing failures in llm agent trajectories via dependency-guided search. *arXiv preprint arXiv:2606.00765*, 2026.

Earl D Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2):115–135, 1974.

Jiaming Wang, Ziteng Feng, Jiangtao Wu, Ruihao Li, Qianqian Xie, Yuxiang Ren, He Zhu, Xueming Han, Fanyu Meng, Junlan Feng, et al. Where do deep-research agents go wrong? span-level error localization in agent trajectories. *arXiv preprint arXiv:2606.02060*, 2026a.

Minxing Wang, Xiaofei Xie, and Yintong Huo. Trajaudit: Automated failure diagnosis for agentic coding systems. *arXiv preprint arXiv:2605.26563*, 2026b.

Yawen Wang, Wenjie Wu, Junjie Wang, and Qing Wang. From flat logs to causal graphs: Hierarchical failure attribution for llm-based multi-agent systems. *arXiv preprint arXiv:2602.23701*, 2026c.

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

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

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

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, et al. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*, 2025.

A Implementation Details

Additional Setup for Main Experiments. We evaluate our method with three LLMs of different scales: Qwen3-32B (Yang et al., 2025), GPT-5.4-mini (OpenAI, 2026b), and GLM-5.2 (Zeng et al., 2026), to assess its robustness across model scales. All LLM calls use a temperature of 0 to reduce generation stochasticity. For *Adaptive Trajectory Retention*, we set the minimum retention ratio $\rho_{\min}$ to 0.25 and the full-retention threshold N_0 to 24. In the main experiments, we set the maximum number of zoom calls K to 8. We report the mean performance over three independent runs to mitigate run-to-run variation.

Additional Setup for Self-Improvement. We conduct experiments on three suites of τ^2-Bench: *airline*, *retail*, and *telecom*, using 114 sampled tasks from the full *telecom* suite. For the *Offline Experience* and *Harness Optimization* settings, we split the tasks at a 1:2 ratio into experience-construction and evaluation sets. For both online and offline settings, we compare four approaches: (1) **Vanilla**, which directly analyzes the failed trajectory to identify a critical error step and generate feedback; (2) **Self-Reflection**, which first reflects on the trajectory before generating feedback; (3) **TrajDebug** (Qi et al., 2026); and (4) our DEAN. We then evaluate how the resulting feedback supports agent self-improvement under the two settings. The prompts for **Vanilla**, **Self-Reflection**, and experience construction in the *Offline Experience* setting are provided in Appendix B. For **TrajDebug** and DEAN, we use the identified failure region under the *Tol.@2* criterion as the basis for generating feedback.

Sensitivity Analysis. DEAN's trajectory compression is governed by two hyperparameters of *Adaptive Trajectory Retention*: the minimum retention ratio $\rho_{\min}$ and the full-retention threshold N_0. Their design follows a simple principle: retain the full trajectory when it is short enough to process cheaply, and once compression becomes necessary, smoothly decay the retention ratio toward a moderate floor. This preserves sufficient global signal to orient the search while reducing the overview size for longer trajectories. As shown in Table 4, the adopted $\rho_{\min} = 0.25$ and $N_0 = 24$ provide a favorable balance between localization accuracy and token cost. Lowering $\rho_{\min}$ below 0.25 progressively degrades localization, most notably in Tol.@2, as aggressive compression removes coarse error, status, and result signals that help guide the global search. In contrast, increasing $\rho_{\min}$ beyond 0.25 yields only marginal accuracy gains while incurring substantially higher token costs, since a larger overview is re-fed at each search turn. The choice $\rho_{\min} = 0.25$ therefore retains sufficient global signal without unnecessarily expanding the context.

Table 4: **Sensitivity analysis**. of the two *Adaptive Trajectory Retention* hyperparameters, averaged over the seven benchmarks and three independent runs; input tokens are reported in thousands (K).

Value	Acc.	Tol.@2	Input Tok. (k)
(a) Minimum retention ratio $\rho_{\min}$			
0.10	19.45	50.29	78.4
0.15	21.75	52.36	92.6
0.25	22.78	55.54	106.8
0.35	22.78	55.70	128.2
0.50	23.01	55.47	165.5
(b) Full-retention threshold N_0			
8	21.17	52.13	98.1
16	22.21	54.43	104.2
24	22.78	55.54	106.8
32	22.90	55.47	110.1
48	20.14	52.13	117.9

The effect of N_0 is considerably smaller: performance remains largely stable for $N_0 \geq 16$ and drops only when compression is triggered too early on medium-length trajectories. This is consistent with its role in determining when compression begins rather than directly controlling the fidelity of the retained overview. Setting $N_0 = 24$ thus preserves full information for short trajectories while placing longer trajectories in the decaying-retention regime. Overall, these results suggest that the selected hyperparameters offer a practical balance between localization quality and overview size within the evaluated settings.

B PROMPTS

System Prompt

```
You are a trajectory forensics analyst.

A FAILED agent trajectory is given.  Find the single CRITICAL ERROR
STEP: the earliest decisive step whose mistake is NOT later repaired
and that leaves an impact on the final failure | fixing that one
step (with everything before it unchanged) would have turned the
failure into a success.  Many steps look wrong; most are later
repaired or harmless.  Commit to ONE step.  Your final answer is
a single best-guess step index m.  Do NOT hedge with a wide interval
| pick the single most likely origin step.

Search discipline (do not fixate on your first suspicion):

- Each turn, form ONE best-guess region and zoom it (pick the single
most promising region from everything you have seen).  Go wide
first, then narrow.

- If a zoomed window shows no decisive un-repaired terminal-impact
error, do NOT answer and do not keep narrowing it | next turn zoom a
DIFFERENT region.

- Once a window contains the decisive origin, narrow inside it
to the single step where the wrong decision was first made, then
answer.

- Your answer step MUST lie within a window you zoomed; never answer
purely from the overview.

- The critical step is NOT necessarily mid/late | consider the whole
trajectory, including early steps.  Prefer the earliest un-repaired
origin over later, louder errors.

You are given an overview of the trajectory.  Call zoom_in(l, r) to
inspect the inclusive window [l, r] at higher detail.  You have at
most {max_zoom} zoom_in calls.

Each zoom_in response includes an "analysis" field written by a
per-window analyst:

{analyst_rubrics}

The origin can be in ANY position of the trajectory | EARLY, MID,
or LATE. Do NOT fixate on one suspect region or keep re-zooming the
same area | if a region returns symptom/clean, move to a DIFFERENT,
still-unexamined part of the trajectory.  Waste no window:  every
zoom should cover territory you have not yet inspected.

When confident, stop calling tools and output your single best-guess
step in exactly one form:

<answer>m</answer> # the single most likely critical step index
```

Global Trajectory Prior Construction Prompt

```
Read the trajectory and produce three memory documents.

Focus on what is SURPRISING, SUSPICIOUS, or INCONSISTENT --- not on
confirming expectations.

===HIGH_LEVEL=== 4-6 bullets:  what the task asks, what the agent
answered, overall strategy, what looks wrong at a glance, which part
did the most critical work.

===MID_LEVEL=== One entry per KEY MOMENT (skip routine steps).
```

```
'### Step N' + what happened & why it matters (2-3 sentences).

Include a step if it:  - computed/chose a value feeding the
final answer, - made a direction-changing decision, - looks
wrong/surprising, - introduced NEW information, or - matches an
error signal.

===LOW_LEVEL=== One entry per SUSPICIOUS/CRITICAL step.

'### Step N' +:  exact values, data provenance, completeness,
assumptions, what looks wrong/risky, inputs received & whether
correct.
```

Segment-centric Reasoning Prompt

```
You are a per-WINDOW forensics analyst for a FAILED agent
trajectory, given the GLOBAL memory M, ONE inspected window, and the
TERMINAL FAILURE. Decide whether THIS window contains the CRITICAL
ERROR STEP.

To judge THIS window:

(1) Find any step where the agent makes a WRONG ASSUMPTION/ACTION
of the kind above | check it against the environment's response and
against facts the agent already observed (contradiction -> wrong).

(2) Check whether it is repaired by any later step in M.

(3) Use M's LOW_LEVEL to check whether an EARLIER step already
introduced the same wrong assumption/action (then this window is
a symptom).

Output EXACTLY these labeled lines, terse, no preamble:

AGENT_BEHAVIOR: the concrete action/assumption in this segment (1-2
sentences).

ANALYSIS: is this the FIRST un-repaired wrong assumption/action
(cite the environment response or observed fact it contradicts)?  or
reasonable / downstream symptom?  (1-2 sentences).

FAILURE_LINK: does it propagate to the terminal failure, or clean?
(1 sentence)

VERDICT: one of {cause_candidate, symptom, clean} | output ONLY the
label word:

- cause_candidate:  an un-repaired wrong assumption/action made HERE
could reach the final failure.

- symptom:  the wrong assumption/action was introduced EARLIER.

- clean:  no decisive error here.

Do NOT output a step number/interval.  Do NOT suggest where to zoom
next.
```

Vanilla Prompt

```
Trajectory:  trajectory

Your task:

1.  Identify the earliest step which directly leads the agent off
track or repeats ineffective behaviour.
```

```
2. Reference that exact step number as shown in the trajectory. Do
not shift to later steps of that error.

3. Explain why the chosen step is wrong, citing relevant
observation/action details.

4. Suggest a concrete alternative for that same step that would
move the agent toward success (e.g., a specific action to take
instead).

Respond strictly in the following format (single spaces around
colons, no extra text):

step:<number>

reason:<one concise, specific sentence>

suggestion:<one actionable suggestion for that step>
```

Self-Reflection Prompt

```
Current result: {trajectory}

Why is this trajectory not finished the task?

Feedback:
```

Experience Construction Prompt

```
Below is analysis from several FAILED attempts. Read ALL of it,
then DISTILL a concise, GENERALIZED experience list for a fresh
agent working on NEW tasks: {analysis}

Include the recurring pitfalls to avoid and the corrective
approaches to take. Do not include instance-specific IDs or booking
numbers; generalize across tasks.

FEEDBACK:
```

C Additional Results

Method	AgentErrorBench						Who&When				TrajErrBench				Avg.	
	ALFWorld		GAIA		WebShop		Hand-Crafted		Algorithm		τ^2-Bench		SWE-Bench Pro			
	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Avg. Traj. Length	60.00		28.68		51.84		51.60		8.72		29.27		119.70		49.97	
							Qwen3-32B									
Direct Prompting	58.43	0.01	45.06	0.01	45.63	0.01	19.55	0.01	4.82	0.01	8.06	0.01	50.70	0.01	22.62	0.01
AgentDebug	330.10	15.95	898.04	8.70	248.21	13.82	19.93	0.34	4.70	0.36	3.37	0.40	17.86	1.32	108.24	3.51
AgentRX	1439.12	5.61	361.16	2.85	1015.14	6.09	443.55	5.35	101.96	5.09	176.26	5.86	1277.73	5.27	496.35	5.47
CHIEF	419.99	9.90	86.71	2.08	319.84	14.34	140.17	7.58	33.88	4.75	66.65	10.62	365.27	11.36	152.71	9.29
TrajDebug	755.51	50.44	527.35	32.79	648.19	51.61	852.21	33.80	112.03	7.58	226.30	21.75	1375.96	46.07	467.43	28.55
DEAN	211.89	2.91	63.92	1.73	138.29	2.62	85.34	2.46	65.92	2.29	90.79	2.74	139.83	2.52	106.78	2.59

Table 5: Computational overhead across multiple benchmarks. All token counts are reported in thousands (K).

D Algorithm

Algorithm 1 summarizes the full DEAN procedure of Section 3. Phase 1 performs model-free trajectory compression; Phase 2 builds the global trajectory prior; Phase 3 performs the global-to-local *Thinking with Trajectories* search. The per-window analyst π_{analyst} is instantiated with the *Segment-centric Reasoning Prompt*, the search driver π with the *System Prompt* (Appendix B). Hyperparameters $\rho_{\min}$ and N_0 follow Appendix A.

Algorithm 1 Algorithm of DEAN.

Require: Instruction I for global-to-local search; Instruction I_{analyst} for segment-centric reasoning; Failed trajectory $\tau = \{s_1, \ldots, s_N\}$; LLM π; Max turns X.
Ensure: Critical failure step t^*.

▷ **Phase 1: Trajectory Compression** (model-free)

$\rho \leftarrow \rho_{\min} + (1 - \rho_{\min}) \exp\Big(-\frac{(N-N_0)_+}{N_0}\Big), \quad B \leftarrow \lfloor \rho \cdot |\tau| \rfloor$ ▷ *Adaptive retention*

$B_i \leftarrow B \frac{\log n_i}{\sum_{j=1}^{N} \log n_j}, \quad \forall i \in \{1, \ldots, N\}$ ▷ *Step-wise budget allocation*

for $i = 1$ **to** N **do**
  if s_i is an agent step **then**
    $\tilde{s}_i \leftarrow \text{compress}_{\text{agent}}(s_i, B_i)$
  else if s_i is an environment step **then**
    $\tilde{s}_i \leftarrow \text{compress}_{\text{env}}(s_i, B_i)$
  end if
end for

$\mathcal{T} \leftarrow \{\tilde{s}_i\}_{i=1}^{N}$ ▷ *Role-aware compressed trajectory*

▷ **Phase 2: Global Trajectory Prior**

Extract hierarchical context $\mathcal{H} \leftarrow (\mathcal{H}_{\text{hi}}, \mathcal{H}_{\text{mid}}, \mathcal{H}_{\text{lo}})$ from $\mathcal{T}$ with an LLM

Initialize message history $\mathcal{M} \leftarrow \emptyset$

▷ **Phase 3: Global-to-Local Search**

for $x = 1$ **to** X **do**
  $(\mathcal{O}, [\ell, r]) \leftarrow \pi(I, \mathcal{T}, \mathcal{M})$ ▷ LLM autonomously selects a window
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{O}, \text{zoom}(\ell, r)\}$ ▷ Update messages
  $\mathcal{S} \leftarrow \tau_{\ell:r}$ ▷ Extract target segment
  $e \leftarrow \pi(I_{\text{analyst}}, \mathcal{H}, \mathcal{S})$ ▷ *Segment-centric reasoning*
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{e\}$ ▷ Update messages
  if π commits to a single critical step t given current evidence **then return** $t^* \leftarrow t$
  end if
end forreturn $t^* \leftarrow \pi(\textit{Answer} \mid I, \mathcal{O}, \mathcal{M})$ ▷ Fallback: deduce critical failure step