Inference-Time Scaling of Verification: Self-Evolving Deep Research Agents via Test-Time Rubric-Guided Verification

Yuxuan Wan[†], **Tianqing Fang**[‡*], **Zaitang Li**[‡], **Yintong Huo**[††], **Wenxuan Wang**[‡‡], **Haitao Mi**[‡], **Dong Yu**[‡], **Michael R. Lyu**[†]

[†]The Chinese University of Hong Kong, [‡]Tencent AI Lab
[††]Singapore Management University [‡‡]The Renmin University of China

https://github.com/Tencent/CognitiveKernel-Pro
https://github.com/yxwan123/DeepVerifier

Abstract

Recent advances in Deep Research Agents (DRAs) are transforming automated knowledge discovery and problem-solving. While the majority of existing efforts focus on enhancing policy capabilities via post-training, we propose an alternative paradigm: self-evolving the agent's ability by iteratively verifying the policy model's outputs, guided by meticulously crafted rubrics. This approach gives rise to the **inference-time scaling of verification**, wherein an agent self-improves by evaluating its generated answers to produce iterative feedback and refinements. We derive the rubrics based on an automatically constructed DRA Failure Taxonomy, which systematically classifies agent failures into five major categories and thirteen sub-categories. We present **DeepVerifier**, a rubrics-based outcome reward verifier that leverages the asymmetry of verification and outperforms vanilla agent-as-judge and LLM judge baselines by 12%–48% in meta-evaluation F1 score. To enable practical self-evolution, **DeepVerifier** integrates as a plug-and-play module during test-time inference. The verifier produces detailed rubric-based feedback, which is fed back to the agent for iterative bootstrapping—refining responses without additional training. This test-time scaling delivers 8%–11% accuracy gains on challenging subsets of GAIA and XBench-DeepResearch when powered by capable closed-source LLMs. Finally, to support open-source advancement, we release DeepVerifier-4K, a curated supervised fine-tuning dataset of 4,646 high-quality agent steps focused on DRA verification. These examples emphasize reflection and self-critique, enabling open models to develop robust verification capabilities.

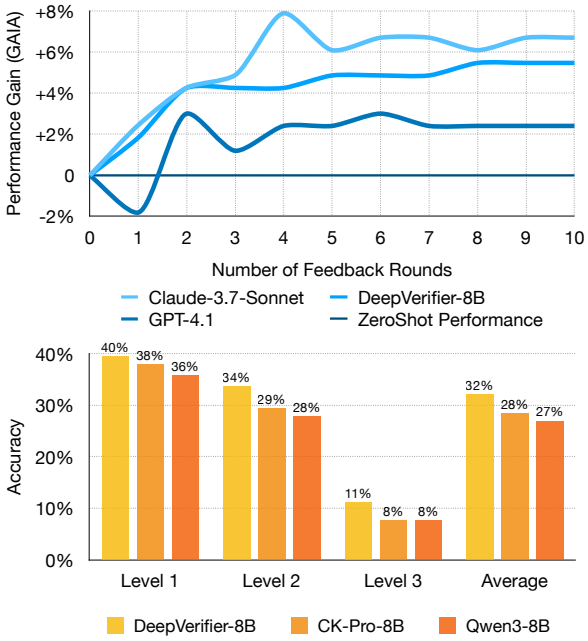

Figure 1: Upper: Inference-time scaling of verification on the full GAIA development set ($n = 165$). Lower: Performance comparison between DeepVerifier-8B fine-tuned on our dataset and other open-sourced models after 10 rounds of verification & feedback on the full GAIA development set.

1 Introduction

Recent advances in Deep Research Agents (DRAs), powered by large language models (LLMs) and vision-language models (VLMs), are transforming automated knowledge discovery and complex problem-solving. These systems demonstrate strong performance on tasks requiring coding, web navigation, file processing, and multi-step reasoning.

However, DRAs remain prone to unreliable outputs stemming from incorrect actions, API failures, hallucinations, or other errors (Song et al., 2025; Li and Waldo, 2024), which significantly constrain their practical deployment (Zhang et al., 2025a). For instance, when tasked with identifying a re-

*Correspondence: yxwan@link.cuhk.edu.hk, tianqfang@tencent.com

Findings of the Association for Computational Linguistics: ACL 2026, pages 24822–24835
July 2-7, 2026

searcher's earliest publication, an agent might rely on incomplete secondary sources and deliver an inaccurate result. In long-horizon tasks involving dozens of pages and hundreds of actions, online human supervision becomes infeasible.

These challenges underscore the need for scalable, automated methods to enhance DRA reliability and performance at **test time** (Zhu et al., 2025c; Hu et al., 2025a). Prior work on inference-time improvement has largely emphasized scaling output tokens or selection across parallel rollouts. For example, Zhu et al. (2025c) introduced parallel sampling for optimal trajectory search, while Gonzalez-Pumariega et al. (2025) employed narrative-driven aggregation across iterations. Despite existence of Reflexion (Shinn et al., 2023)-based methods use textual feedback (Zhou et al., 2025a; Yuksekgonul et al., 2024) to bootstrap the agent response, the generation of feedback itself is a hard task that requires sophisticated reasoning capability (Team et al., 2025; Hu et al., 2025a).

A more robust test-time self-evolution pipeline, wherein an agent iteratively improves its outputs through verification and feedback *without additional training*, involves (1) verifying generated outputs, (2) producing targeted feedback upon detecting errors, and (3) iterating with this feedback. In this paper, we advance this pipeline in two key areas.

For (1) verification, we exploit the **asymmetry of verification** to decompose complex problems into simpler sub-tasks, where checking correctness is often easier than generation (Wei, 2025). For (2) feedback generation, we incorporate rubrics-based rewards (Gunjal et al., 2025; Huang et al., 2025) to provide structured, discriminative signals, derived from an automatically constructed **DRA failure taxonomy**. We construct the the taxonomy by analyzing the failure trajectories on the WebAggregator dataset (Wang et al., 2025), categorizing failures into five major classes and thirteen sub-classes. Based on (1) and (2), we present **DeepVerifier**, an agentic pipeline for automatically verifying the success of DRA output and provide feedbacks based on the rubrics. DeepVerifier decomposes intricate verification challenges into verifiable information-retrieval sub-tasks (Figure 2), overcoming limitations of prior holistic judging approaches. This decomposition principle extends naturally to report generation (Fan et al., 2025). We evaluate DeepVerifier on the GAIA benchmark (Mialon et al., 2023), which assesses core abilities including reasoning, multimodality, web browsing, and tool use. Results show DeepVerifier outperforming vanilla agent-as-judge and LLM judge baselines by 12–48% in meta-evaluation F1 score. When integrated for test-time scaling with capable closed-source LLMs (e.g., Claude-3.5-Sonnet), it yields 8–11% accuracy improvements across challenging GAIA subsets and 3–6% improvements on the XBench-DeepSearch dataset.

Beyond test-time inference, we extend DeepVerifier to develop DeepVerifier-4K, a high-quality supervised fine-tuning (SFT) dataset comprising 4,646 prompt-response pairs tailored for DRA verification. Curated by filtering and parsing 400 initial agent verification trajectories, DeepVerifier-4K enables robust reflection and self-critique. Using this dataset, we fine-tune DeepVerifier-8B, a model that surpasses other open-sourced models after reflection on key benchmarks. Our framework thus offers a scalable solution for both DRA verification and high-quality dataset creation. Moreover, as reflection-enhanced reinforcement learning gains momentum (Hübotter et al., 2026; Liu et al., 2025), our taxonomy and dataset can serve as a foundation for reliable self-verification and reward signals in RL-based agent training. In summary, our contributions are as follows:

- We formalize the agent reflection pipeline for Deep Research Agents (DRAs) and leverage the asymmetry of verification to achieve superior meta-evaluation performance.
- We introduce a comprehensive DRA failure taxonomy, automatically constructed to categorize failures systematically, and derive structured rubrics for outcome-based rewards.
- Through extensive experiments, we demonstrate the inference-time scaling of verification that holds for both capable closed-source LLM APIs and supervised fine-tuned models; integrating enhanced verification capabilities significantly boosts overall agent performance.

2 Related Work

2.1 Deep Research Agents

Research on DRA has rapidly advanced, aiming to build autonomous systems capable of multi-step tasks such as web navigation, data analysis, code generation, and report synthesis. Proprietary frameworks like OpenAI's Deep Re-

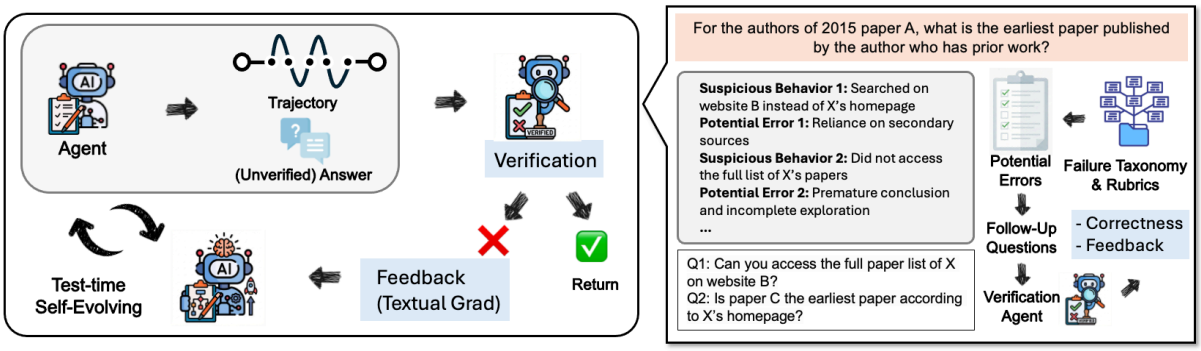

Figure 2: Overview of DeepVerifier, which decomposes complex verification problems into smaller, simpler sub-questions leveraging the asymmetry of verification, and provides corrective feedback for the DRA to retry when the answer is considered incorrect.

search (OpenAI, 2025), Google's Gemini Deep Research (Google DeepMind, 2025), Perplexity's Deep Research (Perplexity AI, 2025), and Moonshot AI's Kimi-Researcher (Moonshot AI, 2025a,b) demonstrate strong performance on benchmarks such as GAIA and Humanity's Last Exam, setting high standards for autonomy and multimodal reasoning (Mialon et al., 2023; Phan et al., 2025; Zhang et al., 2026). Meanwhile, open-source frameworks democratize agent development. Notable systems include SmolAgents (Roucher et al., 2025), the WebAgent family (Wu et al., 2025a; Li et al., 2025; Tao et al., 2025), OWL (Hu et al., 2025b), OAgents (Zhu et al., 2025a,c), and CK-Pro (Fang et al., 2025b), among others (Wu et al., 2025b; Bahdanau et al., 2024; Tang et al., 2025; Zhang et al., 2024; Fang et al., 2025a). While most efforts are being put into Agent Foundation Model Training using Supervised Finetuning (OpenAI, 2025; Hu et al., 2025b; Wu et al., 2025a; Hu et al., 2025c) and Reinforcement Learning (Li et al., 2025, 2026; Yu et al., 2025; Wang et al., 2026; Fang et al., 2026a,b; Zhu et al., 2026), DRA verification and its scaling effect remain underexplored.

2.2 Test-Time Scaling of Agents

Many works apply Test-Time-Scaling (Choi et al., 2023; Snell et al., 2024) to enhance the quality of agent responses. (Zhu et al., 2025b) proposes Best-of-N selection, majority vote, etc. However, such test-time-scaling methods remain prone to the same set of failures in different roll-outs, meaning that errors arising in one run also tend to recur in other runs, rendering the overall result unreliable. Other works explored using LLMs or agents as judges to evaluate agent responses (He et al., 2024; Pan et al., 2024; Lù et al., 2025; Zhuge et al., 2024; Yang et al., 2025). However, these works have focused on web navigation tasks, general reasoning tasks, or software development tasks, while none have studied the responses of DRAs.

Recent research also investigates self-evolving LLMs (Zhou et al., 2025b; Zhang et al., 2025b; Zuo et al., 2025; Zhang et al., 2025a; Feng et al., 2025). For example, recent methods explore code-as-task self-play (Zhou et al., 2025b), self-aware RL (Zhang et al., 2025b), and test-time RL (Zuo et al., 2025), but none address DRAs. (Zhang et al., 2025a) systematically analyze failure modes of DRAs, but do not provide an automated framework for detecting failures or improving agents based on these findings. In contrast, we (1) construct an agent failure taxonomy, (2) introduce a verification-asymmetry–based framework to automatically detect failures, and (3) extend it to self-evolving verification, demonstrating a clear verification scaling effect.

3 DRA Failure Taxonomy

To exploit the asymmetry of verification and decompose complex problems into simpler sub-tasks, we first investigate the common failures of DRA and construct a DRA Failure Taxonomy. To avoid data leakage or contamination and ensure generalization, we select the WebAggregatorQA dataset to construct the taxonomy, and evaluate the framework on three distinct dataset: GAIA, BrowseComp, and XBench-DeepSearch to demonstrate the effectiveness and generalization of the method.

Trajectory Collection To construct the taxonomy, we first collect problem-solving trajectories from a representative deep research agent. Table 1 summarizes the resulting corpus, which is substan-

Table 1: Statistics of collected trajectories. Steps refers to the actions (planning, searching, clicking, etc.) performed by agents and sub-agents. Number of tokens is calculated by the GPT-4o tokenizer.

Trajectory Stat	Min	Max	Avg	Total
Steps	2.0	156.0	33.3	2,997
Tokens	18.7K	60.0M	8.2M	738M
Correct/Incorrect	-	-	-	0.96
Unique Tasks	-	-	-	90

tial (2,997 agent actions), diverse (90 distinct tasks; trajectories range from 2 to 156 steps), and nearly balanced (correct/incorrect ratio of 0.96). We use Cognitive Kernel-Pro (Fang et al., 2025b), a high-performing fully open-source multi-module DRA framework, with Claude-3.7-Sonnet as the backbone model due to its strong performance in this setting. Trajectories are generated by running the agent on WebAggregatorQA (Wang et al., 2025), a benchmark that exercises core DRA capabilities including multi-step reasoning, multimodal inputs, web browsing, and general tool-use proficiency.

Error Points Collection For each trajectory that produces an incorrect final answer, we annotate the underlying failure points. We use the human reference solution traces provided by WebAggregatorQA as a grounding signal, and recruit two research staff annotators to independently inspect the agent's execution and identify deviations from the reference reasoning and evidence-gathering process. Each annotator records a set of *error points*, i.e., concrete, localized mistakes such as missing critical evidence, using an invalid source, or misinterpreting an instruction, along with the supporting trajectory step(s). We then reconcile the two annotation sets through a merge procedure: duplicated items are consolidated, and distinct items are retained in the final list. We calculate that on average, 63.0% of the error points of one annotator overlapped with the other's, indicating a relatively high agreement rate between the annotators. This process yields 555 error points. Full annotation guidelines are provided in Appendix A.

Taxonomy Construction To gain further insight into the failures, we construct a taxonomy based on the error points. In particular, we conduct an iterative analysis and labeling process with two annotators with multiple years of AI research experience from our institute. The initial labels are determined by clustering a subset of 50 error points. In each iteration, we construct a new version of the taxonomy by comparing and merging similar labels, removing inadequate categories, refining unclear definitions based on the results of previous iterations, and discussing the results of the last iteration. As a result, we obtain a classification scheme illustrated in Figure 3. The more frequent the subclass, the wider the branch.

Analysis Figure 3 shows that DRA failures are dominated by Finding Sources, with the largest flows corresponding to errors such as consulting the wrong evidence and relying on generic searches, highlighting that upstream information acquisition is the most frequent point of collapse. Reasoning failures are the next most common, driven by premature conclusions, misinterpretation, and hallucinated or overconfident claims, indicating that even when information is present, agents often make incorrect inferential leaps. Problem Understanding, Action Errors, and Max Step Reached account for the remaining failures, often cascading from early mistakes into long, unproductive trajectories.

4 DeepVerifier

We present an overview of the DeepVerifier framework in Figure 2. We adopt a three-stage multi-module framework in our agent implementation. This framework consists of a decomposition agent, a verification agent, and a judge agent. The following sections describe each module in detail.

4.1 Decomposition Module

The decomposition agent leverages previous trajectories and the DRA failure taxonomy to exploit the asymmetry of verification. Instead of asking the verification agent to re-solve the entire complex task (e.g., "Given a query, an unverified answer, and the agent's trajectory, verify the correctness of the answer"), which often results in high error rates similar to those of the original agent execution, the decomposition agent breaks the problem into smaller, more manageable sub-questions. These sub-questions target specific vulnerabilities in the previous solution, such as "Does source X state claim Y?" or "What is the exact figure for Y in the latest report X?" The workflow of the decomposition agent comprises three steps.

Trajectory Summarization. Agent trajectories average 8.2M tokens, far exceeding any LLM's context window. Moreover, concise descriptions of

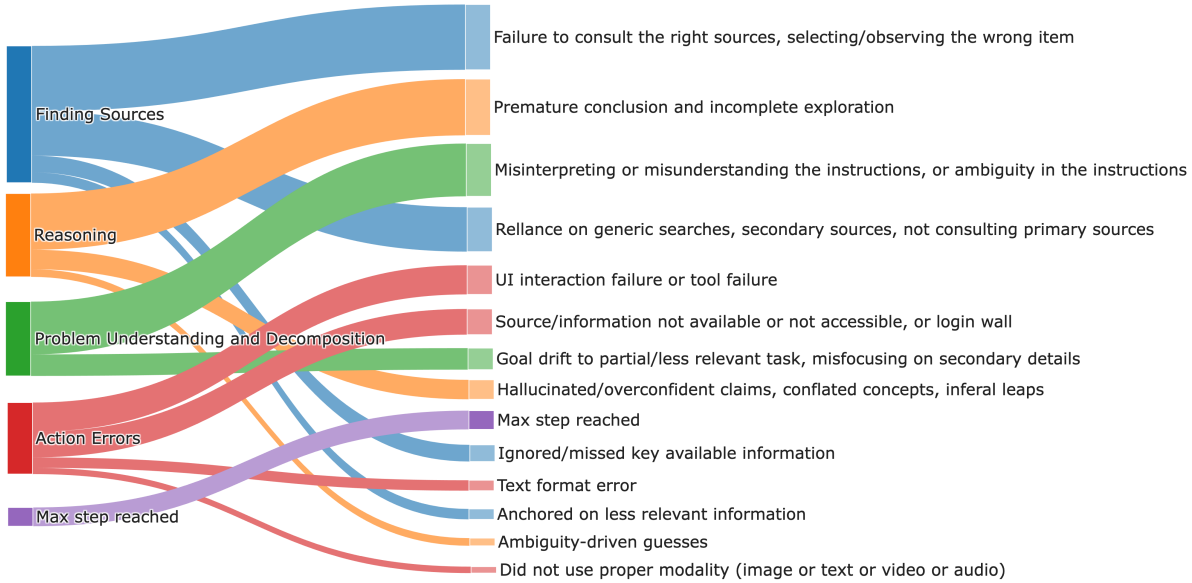

Figure 3: DRA failure taxonomy that categorizes 555 agent failures into five major classes and thirteen subclasses.

rollout steps can improve test-time scaling (Fang et al., 2025b; Gonzalez-Pumariega et al., 2025). We therefore instruct the decomposition agent to first produce a compact, step-indexed synopsis of the trajectory. For each step, it records the source visited and the concrete information retrieved (facts, numbers, quotes). The summary is descriptive, not interpretive, enabling downstream checks without reloading the full trace.

Potential Error Identification. Given the summary and our failure taxonomy in the system prompt, the decomposition agent scans for behaviors that align with known failure modes . It produces paired findings of the form ⟨behavior⟩ ⇒ ⟨potential error + taxonomy label⟩ with a brief justification. These structured pairs localize where and how failures likely arise.

Follow-Up Question Formulation. Finally, the decomposition agent drafts high-leverage follow-up questions targeted at the flagged vulnerabilities. Each question is answerable via external evidence and designed to decisively confirm or refute a risky claim.

By focusing only on essential, potentially faulty claims, this process allows the verification agent to build on well-grounded conclusions, ignore trivial details, and check only for suspicious or unsupported assertions. Detailed prompts of each step are shown in Appendix B.

4.2 Verification Agent and Judge Module

Verification The verification module retrieves answers to the follow-up questions sequentially. In our experiment, we use the CK-Pro agent (Fang et al., 2025b) as the verification agent, a modular multi-agent framework capable of web search, screenshotting, and code execution.

Judge The judge agent evaluates the unverified answer based on the trajectory summary, potential error list, follow-up questions, and their answers. It begins by providing a concise explanation, followed by a score between 1 and 4, where: 1 = entirely incorrect, 2 = mostly incorrect, 3 = mostly correct, 4 = entirely correct.

5 Enhancing Deep Research Agents with Scalable Verification

Test-Time Scaling with Reflection and Feedback. Beyond verification, our framework enhances the test-time scaling performance of DRAs through reflection. By integrating DeepVerifier into the DRA, the agent can review and evaluate its previous actions. Specifically, we modify the judge agent's prompt to: 1) provide actionable instructions for the agent to retry tasks and avoid repeating mistakes, and 2) suggest correct answers if they are already available within the given information (e.g., previous trajectories or follow-up answers). After completing each task, the agent verifies its own outputs using DeepVerifier, collects feedback, and uses it to guide further retries. This process re-

peats until a satisfactory answer is reached or a predefined retry limit is exceeded.

Training Reflection Ability in Agent Foundation Models. Many open-source models, lacking fine-tuning for reflection, show limited test-time scaling capabilities (Fang et al., 2025b). To address this, we propose a deep verification training dataset that leverages existing datasets and DeepVerifier to improve the reflection and test-time scaling abilities of open-source LLMs.

Base Trajectory Collection. We first collect 400 answers and trajectories from agents solving tasks that require significant online exploration and information gathering. These tasks are sampled from the WebAggregatorQA dataset (Wang et al., 2025), which tests agents on information aggregation across 10+ domains. Using the CK-Pro agent with Claude-3.7-Sonnet as the backbone model, we record the answers and corresponding trajectories.

Verification Trajectory and SFT Data Collection. Next, we use DeepVerifier with Claude-3.7-Sonnet to verify the collected base trajectories and answers, saving the verification trajectories. We filter the true positive and true negative verifications—those that correctly accept true answers and correctly reject false ones. After balancing these trajectories, we convert them into prompt-response pairs, resulting in DeepVerifier-4K, a dataset of 4,646 high-quality pairs.

6 Experiment Setup

Models and Benchmarks. We mainly use Claude-3.7-Sonnet as the backbone model of DeepVerifier and other methods. To evaluate the generalization ability of our method, we also compare the performance on GPT-4.1 and Qwen3-8B. We evaluate baselines and our methods primarily on the GAIA-web dataset, which is a subset of the GAIA dataset filtered for tasks that require web browsing following (He et al., 2024). To ensure generalization, we also extend evaluations on the full GAIA dataset (Mialon et al., 2023), XBench-DeepSearch (Chen et al., 2025), and BrowseComp (Wei et al., 2025). XBench-DeepSearch is a Chinese benchmark for search/tool-use, and BrowseComp measures agents' ability to retrieve extremely hard-to-find and entangled information.

Training Configurations. To demonstrate the effectiveness of our approach on open-sourced models, we fine-tune Qwen3-8B on a mixture of DeepVerifier-4K and the CK-Pro-8B training set from (Fang et al., 2025b) to train reflection abilities in open-source models while preserving their foundational capabilities. The training parameters are set as follows:

Table 2: Ablation study on GAIA-Web. "− Verification" corresponds to a decomposition-only (LLM judge) baseline; "− Decomposition" corresponds to a vanilla agent-as-judge baseline (CK-Pro as judge). Metrics are precision/recall of *rejection* (values $\times 100$).

Method	Precision	Recall	Accuracy	F1
DeepVerifier	75.00	**71.43**	**75.56**	**73.17**
- Verification	**100.00**	14.29	60.00	25.00
- Decomposition	86.96	47.62	72.22	61.54

Baselines and Metrics. We use the LLM judge proposed by (Lù et al., 2025) as the LLM verifier baseline, and the CK-Pro Agent (Fang et al., 2025b) as the agent verifier baseline. Detailed prompts are shown in Appendix B. For verification tasks, we calculate the standard precision, recall, accuracy, and F1 score to measure the correctness of the evaluation, where true positive is defined as a verifier assigning "reject" label to a wrong answer, and a true negative is defined as a verifier assigning "accept" label to an correct answer. In the scaling experiment, we treat a score of less than or equal 2 as incorrect, and greater or equal to 3 as correct. We stop the feedback loop as soon as the verifier judge the answer as correct.

Research Questions We investigate the following research questions (RQs) to demonstrate the effectiveness of our method:

1. RQ1: Is DeepVerifier effective in verification?
2. RQ2: Can DeepVerifier help improve the performance of DRA via test-time scaling?
3. RQ3: Can DeepVerifier-4K help improve the reflection ability of open-sourced models?

7 Results & Analysis

7.1 RQ1: Effectiveness of DeepVerifier

We conduct an ablation study using the trajectories of the CK-Pro agent with a Claude-3.7-Sonnet backbone on the GAIA-Web dataset, as described in Table 1. Each method, using the same backbone model, is evaluated on its ability to verify the correctness of these cases. As shown in Table 2, DeepVerifier achieves superior performance across

recall, accuracy, and F1 score. Removing the verification module or decomposition module exhibits high precision (100% and 86.96%, respectively) in detecting erroneous cases, but their recall and accuracy remain unsatisfactory. Closer analysis reveals that these judges are effective at catching obvious mistakes, such as execution failures, but often overlook subtler reasoning or factual errors, accepting many incorrect answers as correct. This limitation arises because removing the verification module renders the judge fail to identify secondary-source dependence, overconfident claims, or hallucinated facts supporting incorrect responses. Meanwhile, removing the decomposition does not affect the judge's access to external sources, but we observe that without proper decomposition, the agent tends to check every step by re-solving the entire task, leaving them vulnerable to the same reasoning errors as the original agent. In contrast, DeepVerifier decomposes complex verification into smaller, targeted sub-questions that directly test specific vulnerabilities, making it more robust against faulty reasoning and unsupported claims.

Answer to RQ1: DeepVerifier is effective in DRA verification, achieving a balanced precision–recall tradeoff and yields a 12% - 48% improvement in F1 score and highest accuracy compared to ablated versions.

7.2 RQ2: Improving the Performance of DRA Via Reflective Test-Time Scaling

We evaluate whether DeepVerifier can enhance the performance of Deep Research Agents through reflective test-time scaling by integrating it into the CK-Pro agent with Claude-3.7-Sonnet and measuring accuracy across feedback rounds on the GAIA dataset. As shown in Table 3, accuracy consistently improves with additional feedback iterations, reaching its peak at the fourth round. This demonstrates that iterative reflection and verification feedback effectively help the agent refine reasoning and correct previous errors.

Performance on the GAIA dataset. The overall accuracy on GAIA-Full increases from approximately 52% to 59%, with peak value reaching 60.1%, marking the best performance gain of 8%. The GAIA-Web subset shows the greatest improvement, rising from 52% to above 62%, with peak value reaching 63.5%, indicating that web-based, retrieval-heavy tasks benefit most from DeepVerifier 's targeted verification and evidence-grounding process. Meanwhile, the reasoning and file-operation subset also exhibits improvement across rounds, demonstrating that the reflective feedback mechanism generalizes beyond web-based scenarios. GPT-4.1 shows a similar trend, improving from 29.5% to 32.5% (best), confirming cross-model generalization (Figure 1).

Performance on other DRA datasets. Results in Table 4 show that the scaling effect remains consistent despite the multi-lingual nature of DeepSearch and the extreme difficulty of BrowseComp: XBench-DeepSearch improves from 41.0 (0 rounds) to 47.0 (best, +6.0), and ends at 44.0 (+3.0 at 10 rounds); BrowseComp improves from 5.0 to 10.0 (best, +5.0), and ends at 9.0 (+4.0).

Analysis of the Scaling Trend Performance typically peaks in early feedback rounds due to our iterative setting and the verifier's imperfect precision and recall. In each round, the verifier enables many incorrect cases to be fixed (incorrect→correct), but also occasionally rejects correct answers, causing regressions (correct→incorrect). Table 5 shows that the incorrect→correct transition is stronger but decays quickly, whereas the correct→incorrect transition is weaker but persists across rounds; their interplay produces the observed peak around the fourth round.

Inference Cost. While iterative verification introduces additional compute, DeepVerifier is relatively efficient: the decomposition module narrows verification to $\leq$3 targeted follow-up questions rather than re-solving the full task, accuracy gains peak around round 3–4 enabling practical early stopping, and the loop terminates as soon as the verifier accepts the answer. Compared to broad search-based scaling (e.g., Best-of-N with full re-execution), this yields a favorable accuracy–cost tradeoff without additional training.

Answer to RQ2: DeepVerifier effectively scales DRA performance through structured reflection: as feedback rounds increase, the agent progressively enhances its accuracy, achieving over 8% performance gains on Claude-3.7-Sonnet without additional training or external supervision. The scaling behavior also generalizes to other models and datasets.

Table 3: Accuracy(%) on different subsets of the GAIA dataset with different rounds of feedback using DeepVerifier (DV) across different backbone models.

GAIA Split	Model	# Feedback Rounds						Final Gain	Best Gain
		0	2	4	6	8	10		
Web	Claude-3.7	51.11	58.89	63.33	62.22	61.11	62.22	11.11	12.22
	GPT-4.1	28.89	32.22	31.11	32.22	31.11	31.11	2.22	3.33
	DV-8B	26.67	31.11	31.11	32.22	33.33	33.33	6.67	6.67
File/Reasoning/Others	Claude-3.7	53.57	53.57	56.21	54.92	54.92	54.92	1.35	2.64
	GPT-4.1	30.67	33.33	33.33	33.33	33.33	33.33	2.67	2.67
	DV-8B	26.81	30.85	30.85	30.85	30.85	30.85	4.04	4.04
Full	Claude-3.7	52.22	56.49	60.12	58.93	58.32	58.93	6.71	7.90
	GPT-4.1	29.51	32.53	31.92	32.53	31.92	31.92	2.41	3.01
	DV-8B	26.73	30.99	30.99	31.60	32.21	32.21	5.48	5.48

Table 4: Accuracy(%) across different datasets versus feedback rounds using DeepVerifier with Claude-3.7-Sonnet backbone.

Dataset	0	1	2	3	4	5	6	7	8	9	10	Final Gain	Best Gain
DeepSearch	41.0	42.0	47.0	41.0	45.0	44.0	43.0	44.0	42.0	44.0	44.0	3.0	6.0
BrowseComp	5.0	8.0	10.0	10.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	4.0	5.0

Table 5: Transition rates between consecutive feedback rounds.

Feedback Round	1	2	3	4	5	6	7	8	9	10
Incorrect to Correct Ratio (%)	18.99	9.33	6.94	8.45	0.00	1.45	0.00	0.00	1.45	0.00
Correct to Incorrect Ratio(%)	12.79	4.44	4.30	1.06	3.03	0.00	0.00	1.03	0.00	0.00

7.3 RQ3: Enhancing Reflection Ability of Open-Sourced Models

We further investigate whether incorporating reflection ability through SFT can improve the reasoning and verification performance of Deep Research Agents. We fine-tune Qwen3-8B on a mixture of DeepVerifier-4K and the CK-Pro training set (Fang et al., 2025b), which we name DeepVerifier-8B, and use this model as the backbone for CK-Pro Agent with DeepVerifier as the reflection module, measuring accuracy after 10 feedback rounds on the GAIA dataset. As shown in Figure 1, models fine-tuned with the DeepVerifier-4K dataset exhibit notable performance gains when equipped with reflection. Specifically, DeepVerifier-8B, which is trained with both the CK-Pro dataset and the DeepVerifier-4K reflective data, achieves the highest accuracy of 32.2% after reflection, representing a 5.5% improvement over its non-reflective result. In contrast, CK-Pro-8B, trained only on the CK-Pro dataset, achieves a smaller gain of 2.6 points, while Qwen3-8B, which lacks both CK-Pro and DeepVerifier training, shows minimal improvement.

Answer to RQ3: Incorporating DeepVerifier 's reflection ability through fine-tuning significantly improves the reasoning and verification performance of Deep Research Agents. The fine-tuned DeepVerifier-8B model achieves a 5.5% accuracy gain compared to its non-reflective version and the Qwen3-8B model.

8 Conclusion

In this paper, we address the challenge of silent and repeated failures in Deep Research Agents by systematically leveraging the asymmetry of verification. We construct a human-annotated failure taxonomy, introduce a taxonomy-guided vulnerability localization mechanism that transforms verification from holistic re-solving into targeted evidence checking, and demonstrate consistent improvements across models and datasets. We also release DeepVerifier-4K to empower the open community to build more trustworthy agents. Our framework offers a practical solution for scalable DRA verification, and we believe it can meaningfully aid the growing body of work on reflection-enhanced reinforcement learning for agents.

9 Limitations

Dependence on model capability. DeepVerifier relies on models' ability to follow rubrics precisely, perform careful cross-checking, and express structured feedback. When the underlying model is weak or lacks sufficient tool-use ability, feedback quality can degrade and yield noisy test-time gains. DeepVerifier-4K can help alleviate this limitation for open-sourced models via SFT.

Test-time cost and latency. While DeepVerifier can minimize the redundant problem-solving steps, iterative verification still introduces extra inference steps (and often additional tool calls), increasing runtime and token usage.

Acknowledgments

This research is supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14209124) under the General Research Fund.

References

Dzmitry Bahdanau, Nicolas Gontier, Gabriel Huang, Ehsan Kamalloo, Rafael Pardinas, Alex Piché, Torsten Scholak, Oleh Shliazhko, Jordan Prince Tremblay, Karam Ghanem, Soham Parikh, Mitul Tiwari, and Quaizar Vohra. 2024. Tapeagents: a holistic framework for agent development and optimization. *arXiv preprint arXiv:2412.08445*.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, and 1 others. 2025. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *arXiv preprint arXiv:2506.13651*.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. 2023. KCTS: knowledge-constrained tree search decoding with token-level hallucination detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14035–14053. Association for Computational Linguistics.

Tianyu Fan, Xinyao Niu, Yuxiang Zheng, Fengji Zhang, Chengen Huang, Bei Chen, Junyang Lin, and Chao Huang. 2025. Understanding deepresearch via reports. *arXiv preprint arXiv:2510.07861*.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. 2025a. Webevolver: Enhancing web agent self-improvement with coevolving world model. *arXiv preprint arXiv:2504.21024*.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, Hongming Zhang, Haitao Mi, and Dong Yu. 2025b. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *Preprint*, arXiv:2508.00414.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chaowen Hu, Lu Pan, Ke Zeng, and Xunliang Cai. 2026a. How to allocate, how to learn? dynamic rollout allocation and advantage modulation for policy optimization. *arXiv preprint arXiv:2602.19208*.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chang Liu, and Peilin Zhao. 2026b. Proximity-based multi-turn optimization: Practical credit assignment for llm agent training. *arXiv preprint arXiv:2602.19225*.

Kaituo Feng, Manyuan Zhang, Hongyu Li, Kaixuan Fan, Shuang Chen, Yilei Jiang, Dian Zheng, Peiwen Sun, Yiyuan Zhang, Haoze Sun, and 1 others. 2025. Onethinker: All-in-one reasoning model for image and video. *arXiv preprint arXiv:2512.03043*.

Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. 2025. The unreasonable effectiveness of scaling agents for computer use.

Google DeepMind. 2025. Gemini deep research — your personal research assistant.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Chen Hu, Haikuo Du, Heng Wang, Lin Lin, Mingrui Chen, Peng Liu, Ruihang Miao, Tianchi Yue, Wang You, Wei Ji, Wei Yuan, Wenjin Deng, Xiaojian Yuan, Xiaoyun Zhang, Xiangyu Liu, Xikai Liu, Yanming Xu, Yicheng Cao, Yifei Zhang, and 48 others. 2025a. Step-deepresearch technical report. *Preprint*, arXiv:2512.20491.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. 2025b. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation.

Minda Hu, Tianqing Fang, Jianshu Zhang, Junyu Ma, Zhisong Zhang, Jingyan Zhou, Hongming Zhang, Haitao Mi, Dong Yu, and Irwin King. 2025c. Webcot: Enhancing web agent reasoning by reconstructing chain-of-thought in reflection, branching, and rollback. *arXiv preprint arXiv:2505.20013*.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, and 1 others. 2025. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. 2026. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*.

Eric Li and Jim Waldo. 2024. Websuite: Systematically evaluating why web agents fail. *arXiv preprint arXiv:2406.01623*.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Websailor: Navigating super-human reasoning for web agent.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. 2026. Verified critical step optimization for LLM agents. *CoRR*, abs/2602.03412.

Zihan Liu, Shun Zheng, Xumeng Wen, Yang Wang, Jiang Bian, and Mao Yang. 2025. Deep self-evolving reasoning. *arXiv preprint arXiv:2510.17498*.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stánczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. 2025. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. *ArXiv*, abs/2311.12983.

Moonshot AI. 2025a. Kimi-k2.

Moonshot AI. 2025b. Kimi-researcher: End-to-end rl training for emerging agentic capabilities.

OpenAI. 2025. Introducing deep research. Technical report, OpenAI.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.

Perplexity AI. 2025. Introducing perplexity deep research.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. Smolagents: a smol library to build great agentic systems.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Kevin Song, Anand Jayarajan, Yaoyao Ding, Qidong Su, Zhanda Zhu, Sihang Liu, and Gennady Pekhimenko. 2025. Aegis: Taxonomy and optimizations for overcoming agent-environment failures in llm agents. *arXiv preprint arXiv:2508.19504*.

Jiabin Tang, Tianyu Fan, and Chao Huang. 2025. Autoagent: A fully-automated and zero-code framework for llm agents. *arXiv preprint arXiv:2502.05957*.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Webshaper: Agentically data synthesizing via information-seeking formalization.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, and 1 others. 2025. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*.

Rui Wang, Ce Zhang, Junyu Ma, Jianshu Zhang, Hongru Wang, Yi Chen, Boyang Xue, Tianqing Fang, Zhisong Zhang, Hongming Zhang, Haitao Mi, Dong Yu, and Kam-Fai Wong. 2025. Explore to evolve: Scaling evolved aggregation logic via proactive online exploration for deep research agents.

Tianyi Wang, Long Li, Hongcan Guo, Yibiao Chen, Yixia Li, Yong Wang, Yun Chen, and Guanhua Chen. 2026. Anchored policy optimization: Mitigating exploration collapse via support-constrained rectification. *CoRR*, abs/2602.05717.

Jason Wei. 2025. Asymmetry of verification and verifier's law. Accessed: 2025-10-30.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025a. Webdancer: Towards

autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025b. Webwalker: Benchmarking llms in web traversal. *CoRR*, abs/2501.07572.

Yiliu Yang, Yilei Jiang, Qunzhong Wang, Yingshui Tan, Xiaoyong Zhu, Sherman SM Chow, Bo Zheng, and Xiangyu Yue. 2025. Quadsentinel: Sequent safety for machine-checkable control in multi-agent systems. *arXiv preprint arXiv:2512.16279*.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. 2025. Guided self-evolving llms with minimal human supervision. *CoRR*, abs/2512.02472.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Dingling Zhang, He Zhu, Jincheng Ren, Kangqi Song, Xinran Zhou, Boyu Feng, Shudong Liu, Jiabin Luo, Weihao Xie, Zhaohui Wang, and 1 others. 2025a. How far are we from genuinely useful deep research agents? *arXiv preprint arXiv:2512.01948*.

Hangfan Zhang, Siyuan Xu, Zhimeng Guo, Huaisheng Zhu, Shicheng Liu, Xinrun Wang, Qiaosheng Zhang, Yang Chen, Peng Ye, Lei Bai, and Shuyue Hu. 2025b. The path of self-evolving large language models: Achieving data-efficient learning via intrinsic feedback. *arXiv preprint arXiv:2510.02752*.

Hongming Zhang, Xiaoman Pan, Hongwei Wang, Kaixin Ma, Wenhao Yu, and Yu Dong. 2024. Cognitive kernel: An open-source agent system towards generalist autopilots. *CoRR*, abs/2409.10277.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. 2026. Progresslm: Towards progress reasoning in vision-language models. *arXiv preprint arXiv:2601.15224*.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. 2025a. Self-challenging language model agents. *arXiv preprint arXiv:2506.01716*.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. 2025b. Self-challenging language model agents. In *Advances in Neural Information Processing Systems (NeurIPS 2025)*. NeurIPS 2025 poster.

Bin Zhu, Qianghuai Jia, Tian Lan, Junyang Ren, Feng Gu, Feihu Jiang, Longyue Wang, Zhao Xu, and Weihua Luo. 2026. Marco deepresearch: Unlocking efficient deep research agents via verification-centric design. *arXiv preprint arXiv:2603.28376*.

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, and 5 others. 2025a. Oagents: An empirical study of building effective agents.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. 2025b. Scaling test-time compute for llm agents. *Preprint*, arXiv:2506.12928.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, and 1 others. 2025c. Scaling test-time compute for llm agents. *arXiv preprint arXiv:2506.12928*.

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

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS 2025)*. NeurIPS 2025 poster.

A Annotation Instructions

This instruction is used for the human annotator for summarizing the error points in each erroneous trajectory.

Instruction for Error Points Annotation

You are given a human execution of a task (which is the ground truth) and an LLM agent execution of the same task (which is different from the ground truth). Please compare and explain how LLMs executions are different from human executions, focusing on finding sources, locating information in the source, drawing observations from sources, problem understanding, etc. Then summarize the reasons why the LLM made the errors in bullet points with short sentences based on the comparison.

B Agent Prompts

B.1 Decomposition Module

Trajectory Summary Prompt

Summarize each step in the trajectory. For every step, list the online sources visited by the agent and the key info obtained from each source.

Required format (repeat "Step N" blocks as needed):
Step 1:
Source 1: source visited by the agent
Info 1: information obtained from the source
Source 2: source visited by the agent
Info 2: information obtained from the source
Step 2:

...
Here is the trajectory:
[Trajectory]

Error Identification

Identify suspicious behaviors and map each to **one** potential error from the list below. If none, return exactly: 'No potential errors found'.

Potential error list:
[Failure Taxonomy]

Required format (or the single-line "No potential errors found"):
Suspicious Behavior 1: short description
Potential Error 1: one item from the list
Suspicious Behavior 2: short description
Potential Error 2: one item from the list
...
Here is the trajectory summary:
[Trajectory Summary]

Follow-Up Questions

Assume a web-capable research agent exists. Propose the **fewest** source-question pairs needed to verify 'answer', using 'task', the [Trajectory Summary], and [Potential Errors].

Required format (up to 3 pairs):
Additional Source 1: source
Additional Question 1: a yes-no question based on the source
Additional Source 2: source
Additional Question 2: a yes-no question based on the source
...
Here are the inputs:
[Answer] [Trajectory Summary] [Potential Errors]

B.2 Verification & Judge Module

Verification Agent Prompt

Here is a source and question pair. Answer the question based on the source.
Source: source
Question: question
Return a brief explanation and concise answer to the question based on the source without any additional text.

Judge Agent Prompt

You are given a task description, an unverified answer, a summary of how the agent obtained the unverified answer, and additional answers provided by another research agent regarding the additional questions. Decide if the unverified answer is correct by first providing a concise explanation, then returning a score between 1 and 4, where: 1 = completely incorrect 2 = mostly incorrect 3 = mostly correct 4 = completely correct
Your response should **exactly follow** this format, with no additional content:
Explanation: explanation Score: score

Corrective Feedback Prompt

You are given a task description, a wrong answer given by an agent, a summary of how the agent obtained the wrong answer, and additional answers provided by another research agent regarding the additional questions. Now, the agent will try to solve the task again. Based on these inputs, you need to help the agent retrieve the correct answer by first providing a brief reflection and then providing **no more than three instructions**. Note that 1) the agent will strictly follow your instruction; if it cannot get the correct answer again, which means your instruction is not useful, then you will be punished. 2) point out necessary sources and actions to avoid the agent making the same mistakes again. 3) The agent is good at understanding clear, concise, and accurate instructions rather than long or complex instructions; the latter will confuse it. 4) You can also suggest the answer to the question in the instructions if you can determine the answer from available information. Your response should strictly follow this format without any other content:
Reflection: brief reflection
Instruction 1: instruction Instruction 2: instruction ...

C Verification Rubrics

The agent then assesses both the trajectory and the predicted answer according to the following rubrics derived from the five major failure categories in this taxonomy:

- **Finding Sources:** The agent should consult specific, authoritative sources and avoid relying on generic or secondary evidence.
  - *Excellent:* All key claims are grounded in targeted, high-quality sources directly relevant to the query.
  - *Good:* Most claims are supported by appropriate sources, with minor reliance on secondary references.
  - *Needs Improvement:* Several key claims rely on generic or tangential sources, undermining answer reliability.
  - *Poor:* Frequent use of vague, unverified, or inappropriate sources; critical evidence is missing or unsupported.
- **Reasoning:** The agent's inference chain should be logically sound, free from premature conclusions, misinterpretation, or hallucinated claims.
  - *Excellent:* All conclusions follow directly and correctly from the retrieved evidence, with no overconfident or unsupported claims.
  - *Good:* Reasoning is largely sound, with only minor inferential gaps or slight overstatements.
  - *Needs Improvement:* Noticeable reasoning errors are present, such as premature conclusions or misinterpretation of evidence.
  - *Poor:* Hallucinated claims, contradictory reasoning, or conclusions that conflict with retrieved evidence.
- **Problem Understanding and Decomposition:** The agent should correctly interpret the task and maintain alignment with the original goal throughout execution.
  - *Excellent:* The task is fully and correctly understood; all sub-goals are well-defined and consistently pursued.
  - *Good:* The task is mostly understood, with minor misalignment or unnecessary sub-goal expansion.
  - *Needs Improvement:* Partial misunderstanding of instructions leads to goal drift or incomplete task coverage.
  - *Poor:* The agent fundamentally misinterprets the task, producing outputs irrelevant to the original query.
- **Action Execution:** Each action should be correctly formatted and directed at the appropriate modality or interface.
  - *Excellent:* All actions are executed correctly with no UI, format, or modality errors.
  - *Good:* Actions are mostly correct, with isolated and non-critical execution errors.
  - *Needs Improvement:* Recurring minor errors in action formatting or modality selection hinder progress.

  - *Poor:* Frequent execution failures (e.g., UI errors, wrong tool or modality) that block task completion.

- **Trajectory Efficiency:** The agent should reach a valid answer within a reasonable number of steps, avoiding unproductive loops caused by early errors.
  - *Excellent:* The task is completed well within the step budget, with no unnecessary detours.
  - *Good:* The task is completed within budget, with minor inefficiencies that do not affect the final answer.
  - *Needs Improvement:* Early errors cascade into extended, partially unproductive trajectories, though a result is eventually produced.
  - *Poor:* The step limit is reached without a valid answer, indicating irrecoverable cascading failure.