Graph of Trace: Visualizing Execution Traces of Scientific Agents

Tianci Gao[1,3*] **Haoxuan Li**[1,2*] **Jianhe Li**[4] **Tianxiang Zhao**[1,5]
Runze Shi[2,1] **Weiran Wang**[6] **Zezhao Wu**[2] **Lu Mi**[2,1†]
[1]Shanghai Qi Zhi Institute [2]Tsinghua University [3]Renmin University of China
[4]Beihang University [5]Georgia Institute of Technology [6]Fudan University

Abstract

Scientific AI agents can autonomously carry out complex research workflows, yet these unfolded workflows often remain difficult for humans to inspect and review, limiting interpretable, controllable and effective human–AI collaboration. To address this challenge, we present a monitoring and visualization framework that records fine-grained execution events and organizes them into a directed graph that makes agent workflows explicit as they proceed[1]. The system records intermediate steps (e.g. tool calls and code executions), and renders them as real-time updated visual traces that expose workflow structure. This allows users to examine how results are produced, identify where failures emerge, and better understand agent behavior across different stages of the research process. We conduct an evaluation on complex research tasks with domain experts of interdisciplinary backgrounds in AI, neuroscience, and biology. Experts report that structured traces visualization improves understanding of agent workflows, perceived interpretability, and usability for analysis and further interaction.

1 Introduction

Recent advances in large language models, when coupled with tool use and executable environments, have enabled autonomous agents capable of performing complex scientific research tasks (Ren et al., 2026; Ding et al., 2025). Scientific research is a long-horizon, multi-stage, and highly compositional process comprising multiple interdependent subtasks, such as literature review (Huang et al., 2025b), experimental design (Ren et al., 2026), code implementation (Novikov et al., 2025), tool execution (Ding et al., 2025), data analysis (Chen et al., 2024), and result interpretation (Schmidgall et al., 2025).

However, most existing works pursue an end-to-end framework, aiming at generating the final results or reports without human intervention (Novikov et al., 2025). Errors introduced in early stages can propagate across dependent components, making late-stage correction costly or impractical (Sun et al., 2025). Thus, scientists' involvement also remains essential in these scientific workflows (Bellos et al., 2025): refining research goals, validating assumptions, and ensuring alignment with research quality. To support such involvement, scientists need to maintain a clear understanding of the workflow, review intermediate decisions, and intervene when necessary, rather than only receiving the final report at the end.

To fill this gap, we introduce a modularized monitoring framework that records fine-grained execution events and structures them into an explicit directed graph of the agent's workflow, shown in Fig 1. First, the Scientific Agent carries out research tasks and produces results, and a separate Monitor Agent reconstructs these outputs into normalized structural traces with explicit dependency relations. The resulting trace candidates are then passed to a parser and persistence module, which enforces schema-level constraints and appends only valid nodes and edges to construct *Graph of Trace*. On top of this, the visualization module continuously reflects the evolving graph, allowing users to inspect the research trajectory in real time without interfering with execution itself. By exposing execution traces rather than only outcomes, our approach supports process-level following, reviewing, and timely human monitoring throughout the research process, thereby advancing trustworthy AI-driven scientific research systems. Our work introduces the following contributions:

1. **Conceptual Framework.** We introduce structured execution traces as an abstraction for repre-

*Equal contribution. lihaoxuan@sqz.ac.cn
†Corresponding author. milu@mail.tsinghua.edu.cn
[1]We present our online demo, demo video and open sourced code at https://github.com/NeuroAIHub/Graph-of-Trace-Visualizing-Execution-Trace-of-Scientific-Agents.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 297–306

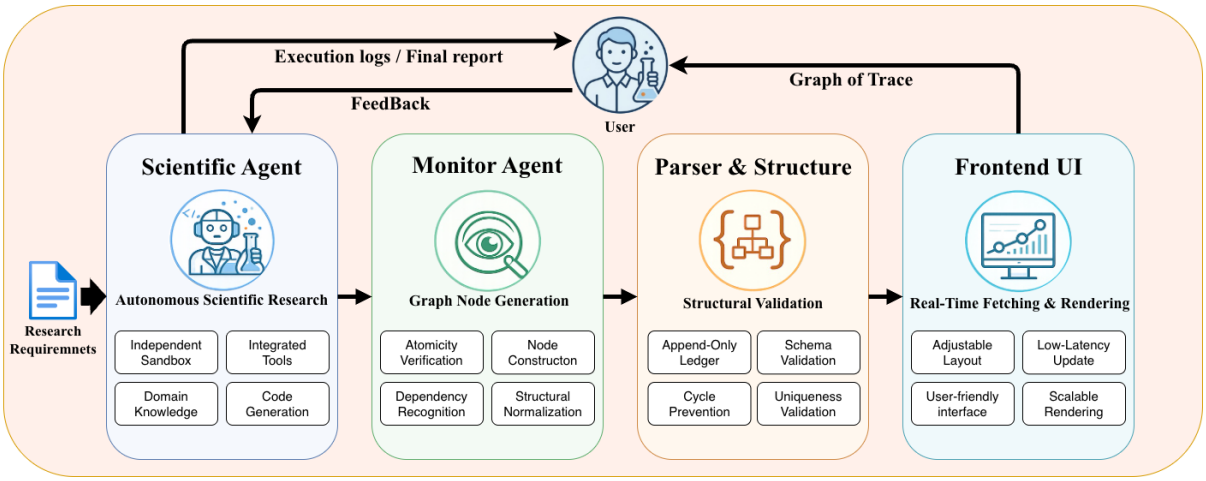

Figure 1: The overview of the framework. The architecture consists of four separate modules: the Scientific Agent performs autonomous task execution; the Monitor Agent receives operations and generates nodes for the garph; the Parser & Structure enforces structural validation; then the Frontend UI renders the *Graph of Trace* for real-time visualization.

senting agent workflows in an interpretable and analyzable form.

2. **System Design.** We develop a monitoring architecture that captures agent actions and constructs a real-time structured execution trace representation.

3. **UI Frontend Design.** We design an interactive interface that visualizes structured traces, enabling intuitive reviewing of execution processes.

2 Related Works

2.1 Reasoning Representation and Visualization Frameworks

Previous work has explored structured representations of LLM reasoning to improve interpretability beyond linear prompt traces. Graph-of-Thought (Besta et al., 2024) generalizes Chain-of-Thought (Wei et al., 2022) and Tree-of-Thought (Yao et al., 2023) paradigms by modeling reasoning as a directed graph of intermediate states, enabling non-linear branching, merging, and reuse of partial reasoning results. Building upon this paradigm, ReasonGraph (Li et al., 2025) provides a web-based interface for visualizing and analyzing reasoning graphs generated by LLMs. DeepRare (Zhao et al., 2026) traces LLM's reasoning process for healthcare tasks to improve the transparency and verifiability of the agent system.

These efforts primarily focus on revealing and inspecting the internal reasoning structure of LLM at the prompt level, instead of the intermediate behavioral actions of agentic systems. Inspired by Graph-of-Thought style visualization, we propose a framework designed to represent and record the full execution process of autonomous scientific agents, instead of revealing the reasoning process.

2.2 Autonomous Scientific Agents and Research Platforms

Recent systems have substantially advanced the autonomy of scientific agents across the research lifecycle. Representative efforts include *The AI Scientist* (Lu et al., 2024), *Robin* (Ghareeb et al., 2025), *SciSciGPT* (Shao et al., 2025), *Biomni* (Huang et al., 2025a), and *OpenLens AI*(Cheng and Suo, 2025), among others. Collectively, this line of work establishes autonomous scientific discovery as a rapidly maturing paradigm which demonstrates the increasing feasibility of long-horizon, tool-integrated, and domain-aware scientific automation, spanning hypothesis generation, experimentation, analysis, and reporting within end-to-end or multi-agent research workflows.

However, most platforms are targeted at exposing final research results without structured, dependency-aware representations of intermediate steps, which limits the capabilities to follow execution trajectories, review intermediate states, and interact via real-time intervention, even though outcome-level evaluation is supported.

3 Framework

3.1 System Overview

For explicitly and correctly visualizing the research process conducted by agents, we design a four-module framework shown in Figure 1, which consists of (1) a Scientific Agent for autonomous execution, (2) a Monitor Agent for reconstructing structural traces, (3) a parser and persistence module that enforces structural contracts, and (4) a visualization module that exposes the evolving *Graph of Trace* in real time.

3.2 Scientific Agent

The Scientific Agent is responsible for task execution and result production. In our implementation, we use OpenHands (Wang et al., 2025) as a representative autonomous agent framework. The agent operates autonomously and reports completed subtasks through MCP calls. It is important to note that the Scientific Agent does not construct or maintain the *Graph of Trace* directly. Instead, the construction of *Graph of Trace* is handled independently by other modules, ensuring that the framework itself is not tightly coupled to any agent platform.

3.3 Monitor Agent

To enable plug-and-play adaptation to any scientific agents or workflows, the Monitor is encapsulated as an MCP-compliant tool (Hou et al., 2026), a protocol-level interface rather than integrating it into the execution framework. The detailed prompt of the agent is attached in the Appendix A.

3.4 Parser and Structure

We define the update process of *Graph of Trace* as an append-only JSON ledger. This append-only design prevents retrospective modifications which could introduce context noises, thus preserving the authenticity of recorded subtasks.

Within this ledger, each node represents a single atomic subtask and contains structured metadata, including a textual description, references to artifacts, and identifiers of the parent nodes. Directed edges capture dependencies between nodes, representing causal or logical progression rather than mere temporal order. Before integrating candidate nodes provided by the Monitor Agent, the Parser enforces schema-constrained validation: nodes must conform to the predefined structure, parent references must exist, and cycles or orphaned dependencies are not allowed. Only nodes and edges that pass validation are appended, maintaining structural integrity and the monotonic growth of the *Graph of Trace*.

3.5 Frontend Visualization

The Frontend Visualization module renders the *Graph of Trace* as a directed graph in which nodes correspond to atomic subtasks and edges encode declared dependencies. Users can inspect node-level metadata, including descriptions, artifacts, and dependency relations, enabling localized auditing of specific execution steps. The visualization module remains synchronized with the persistent *Graph of Trace* state, ensuring consistency between backend structure and frontend rendering while preserving a clear separation between operational execution and interpretative transparency.

4 UI Design

The interface is organized into three coordinated panels to support interaction, monitoring, and inspection of the agent workflow, shown in Figure 2.

The left panel integrates a conversational interface through which users interact directly with the Scientific Agent. In addition to the dialog history, this panel exposes the agent's step-by-step execution process, including actions such as tool invocation, code generation, and command execution. This design allows users to follow the agent's ongoing behavior in a familiar chat-based setting.

The center panel presents the *Graph of Trace*, which is rendered with a React-based[2] frontend. This view visualizes the structural relations among the agent's operations and provides an overview of the workflow as it evolves. Users can freely drag nodes, zoom in and out, and pan across the graph for flexible exploration. A minimap is also provided to support navigation in larger traces.

The right panel provides a detailed inspection for the selected node. When a node is clicked, the interface displays its full operation description, its connected parent nodes, and the corresponding intermediate output produced by the agent at that step. This enables users to examine local execution details while maintaining awareness of the broader workflow structure.

[2] https://legacy.reactjs.org/

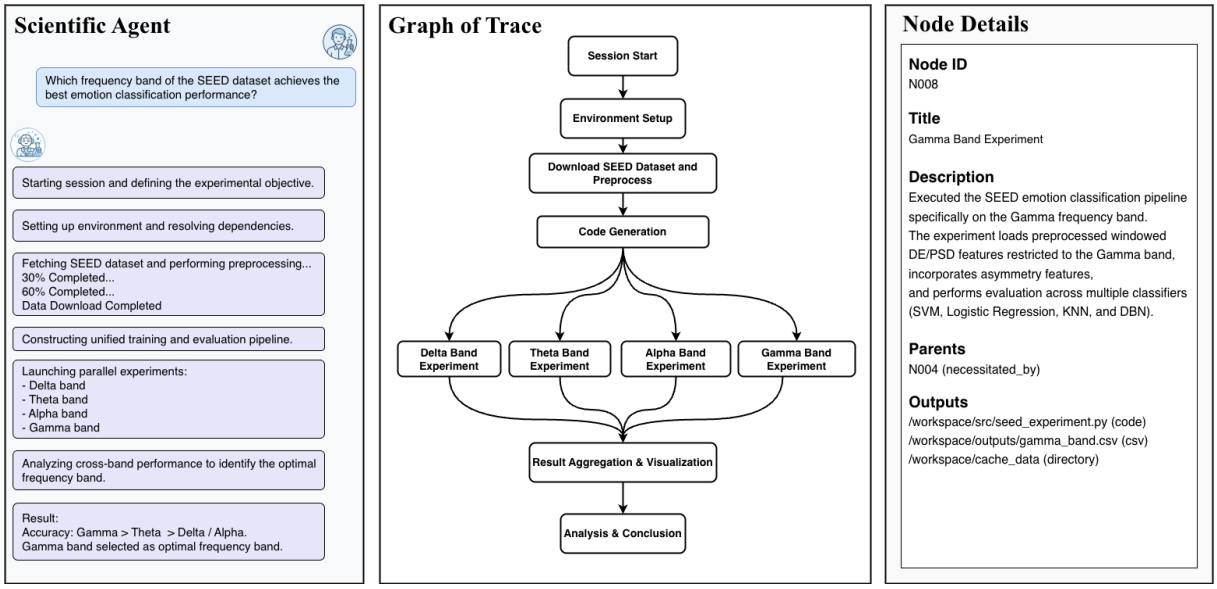

Figure 2: Illustration of UI Design: From left to right: (1) Scientific Agent Conversational Panel. Users interact with the scientific agent through natural language dialogue, while the agent's stepwise responses reflect the progression of the task; (2) *Graph of Trace* Visualization. The agent's workflow is rendered as a layered, top-down directed acyclic graph, beginning with session initialization, followed by environment setup, data preprocessing, model implementation, parallel experiments, and concluding analysis; (3) Node Details. Selecting a node reveals its structured metadata, including identifier, title, description, parent dependencies, and generated intermediate outputs.

5 Evaluation

To evaluate the practical effectiveness of the proposed framework, five domain experts with interdisciplinary backgrounds in AI, neuroscience, and biology conducted a human-centered assessment focusing on real-world scientific replication. Each expert selected one or two representative research papers from their respective domains. They then reproduced the core experiments and workflows using the scientific agent platform integrated with our framework. Overall, we conducted seven case studies of different scientific research that vary substantially in methodological complexity. The generated graphs have diverse structures and different levels of complexity, with the number of nodes ranging from as few as five to more than twenty.

In Figure 3, we have shown *Graph of Trace* in 5 different example scientific research cases. For case (a), the agent is asked to perform a structured exploration of brain computer interface (BCI) Competition IV Dataset under a standardized protocol (Tangermann et al., 2012). The *Graph of Trace* is generally linear in sequence but exhibits parallel operations, resulting in a relatively simple structure. For case (b), it requires the agent to perform analysis like long-term recognition memory effects and representational similarity analysis on the Natural Scenes Dataset (NSD) (Allen et al., 2022). The *Graph of Trace* is also roughly linear in sequence but exhibits more complex parallel and hierarchical operations and dependencies. In case (c), we asked the agent to conduct a simple comparison of machine learning methods which are commonly applied in data-driven scientific discovery. The structure of its *Graph of Trace* diverges into several parallel model evaluation branches, including Logistic Regression, SVM with RBF kernel, and XGBoost. It further coordinates cross-model comparison procedures, generating curve analyses, summary metric tables and comparison figures. For case (d), the agent is required to investigate how reward-related representations in hippocampal neuronal activity evolve over extended experience, based on the predictive coding of reward in the Hippocampus dataset (Yaghoubi et al., 2026). The *Graph of Trace* is more complicated, with more parallel operations and long-range planning. In case (e), we asked the agent to reproduce the baseline motor imagery (MI) experiments from BCI Competition IV (Tangermann et al., 2012). Its *Graph of Trace* has a structure from which we can see a research process proceeding from the initial preparation, to a phase of exploration, then to focused execution, and finally branching into specific directions. Beyond the cases shown in the figure, in case (f), we asked the agent to analyze the SEED dataset

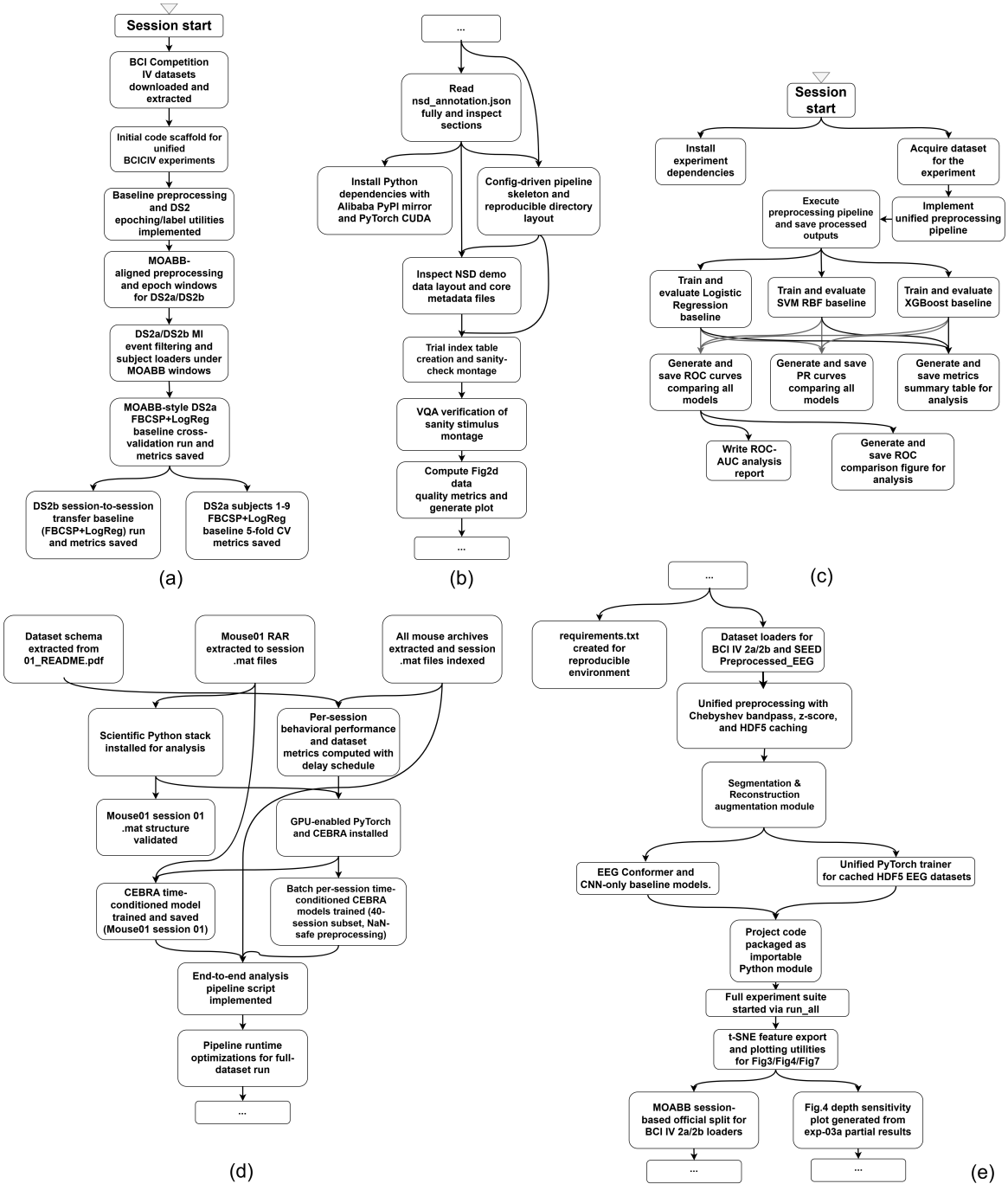

Figure 3: The demonstration of 5 example cases of *Graph of Trace* in the scientific research studies, ranging from neuroscientific discovery to brain-computer interface applications. The selected tasks (a)~(e) include dataset preparation, Python runtime environment setup, data analysis, comparisons of machine learning methods, reproduction of a certain experiment, etc. These tasks vary in methodological complexity, thus resulting in the diversity of structures and complexity of *Graph of Trace*. Note that the block containing an ellipsis (…) represents omitted parts of the execution graph due to space limit.

and identify which frequency band contains more important discriminative information for emotion recognition (Zheng and Lu, 2015). In the *Graph of Trace*, we can observe four parallel branches, each of which trains and evaluates a classification model on a different EEG siganl frequency band (alpha, beta, gamma, and theta). The final node aggregates the results from all branches and concludes that the gamma and theta bands achieve the highest classification accuracy. For case (g), the agent conducted a exploration of the BCI Competition IV Dataset (DS1, DS2a, and DS2b) (Tangermann

et al., 2012), including benchmark evaluation, cross-session transfer analysis, EOG artifact robustness testing, and synthetic data validation. Its *Graph of Trace* follows a mainly linear structure, including dataset preparation, environment setup, preprocessing, multi-dataset experiments, artifact removal comparison, and result visualization. The overall *Graph of Trace* clearly reconstructs the complete research workflow.

We further analyze some failure cases observed in practical use. When the Scientific Agent completes multiple subtasks without immediately reporting them to the Monitor Agent through MCP calls, some operations are not reflected in the *Graph of Trace* in real time. However, the Scientific Agent usually reports the left work in the next MCP call. Therefore, this problem mainly appears as delayed updates rather than permanent information loss.

The second issue is incomplete dependency modeling by the Monitor Agent. In some cases, prerequisite steps such as environment setup or literature review are not correctly identified as parent nodes of downstream experiments. Similarly, during final result summarization, multiple experimental branches may not be fully connected to the final report node, such as task (c) in 3.

The third issue appears when the Scientific Agent starts exploring a new branch after long execution. Because of the long context, the Monitor Agent may fail to identify the correct starting parent node and directly connects the new branch to the last node of the previous branch. This creates incorrect dependencies between branches and makes parallel or independent workflows appear sequential.

Case	Node (Acc)	Edge (Acc)	Lat.	Int.	Aesth.	Usab.	Baseline
1	23/23	30/32	4	2	4	2	0.8
2	9/9	8/8	5	5	5	5	2.0
3	21/21	26/26	5	5	3	4	1.6
4	18/18	19/21	5	4	4	5	4.0
5	5/6	4/5	4	5	5	4	3.2
6	12/13	15/15	5	5	4	5	3.0
7	13/13	18/18	5	5	4	5	3.0
Avg.	0.98	0.96	1.6s	4.43	4.14	4.29	2.51

Table 1: Summary of expert evaluation results across seven experimental cases. Node/Edge (Acc): accurate over total; Lat.: We compute the average latency by taking the midpoint of the selected delay interval and averaging these midpoint values across all cases; Int.: Interpretability; Aesth.: Aesthetics; Usab.: Usability; Baseline: Usability w/o *Graph of Trace*.

Across all cases, we evaluated the quality of the visualization with the following aspects,

Structural Validity. Throughout the evaluation, we observed no formatting or graph rendering errors, indicating that the visualization faithfully preserves the underlying execution structure.

Node-Level Fidelity. The overall node correctness ratio (correct nodes / total nodes) was 0.98, with a mean per-expert accuracy of 0.96 (averaged over individual experts).

Relation-Level Fidelity. The overall edge correctness ratio (correct edges / total edges) was 0.96, with a mean per-expert accuracy of 0.94 (averaged over individual experts).

Temporal Consistency. The system also demonstrated strong consistency with graph updates tightly synchronized to agent execution and a low average latency of 1.6 seconds.

We also collected the users' feedback with the following ratings on a five-point Likert scale (1 = lowest, 5 = highest).

Visual Aesthetics. The interface received an average score of 4.14, noting that the graph remained clear and readable throughout the whole process.

Expert Interpretability. The system achieved a mean rating of 4.43, reflecting that the visualization effectively translated complex agent reasoning into a structured representation that experts could easily follow and review.

Usability. The average score reached 4.29. In contrast, when experts were asked to understand the agent's behavior solely by reading raw dialogues, execution logs, and source code, the average usability rating dropped to 2.51. This comparison indicates that the structured visualization substantially reduced cognitive burden and improved the efficiency of reviewing.

Overall, the evaluation results indicate that in stress tests involving long-horizon, multi-step scientific workflows, *Graph of Trace* enabled real-time auditing of agent behavior. Therefore, the framework effectively bridges raw execution logs and human-interpretable scientific reasoning.

6 Applications

The proposed framework establishes a practical basis for supervising autonomous agent behavior in complex scientific settings and lays the groundwork for more convenient human–agent collaboration. First, the system provides real-time overview for research workflows. By transforming long-horizon sequential execution records into a structured *Graph of Trace*, the framework enables ex-

perts to inspect multi-stage experimental process more clearly rather than just a final report. Furthermore, the explicit presentation of the structured research workflow and its corresponding outputs facilitates the verification and reproducibility of research processes.

7 Conclusion

In this work, we presented a monitoring and visualization framework that establishes a collaborative tracing protocol between the Scientific Agent and a dedicated Monitor. The Monitor Agent then synthesizes these reports into coherent *Graph of Trace* nodes, constructing a real-time, interpretable representation of the research workflow.

Our human-centered evaluation demonstrates that this structured trace visualization significantly enhances expert understanding, perceived interpretability, and usability compared to traditional end-to-end outputs. Ultimately, this framework bridges the gap between autonomous capability and scientific reliability. By facilitating active reporting and structured visualization, we pave the way for more trustworthy, controllable, and collaborative human-AI scientific discovery systems.

In future work, we plan to extend the system with node-level backtracking, enabling experts to revisit earlier states in the workflow and give more precise and in-time intervention while monitoring the agent trajectory in real time, thereby preventing error accumulation across subsequent steps. We also plan to conduct a systematic evaluation of different scientific agents by using structured visualizations to compare how they carry out complex scientific research tasks. It will further support the development of domain-specific benchmarks grounded in observable research trajectories. Finally, we plan to study the interaction between human expert feedback on the *Graph of Trace* and the trajectories produced by agents, with the goal of identifying new patterns of human–AI collaboration in scientific research.

8 Limitations

Despite the promising results, several limitations remain that warrant future investigation.

Reliance on Agent Compliance. The framework assumes that the Scientific Agent consistently invokes the MCP tracing tools at appropriate semantic boundaries. If tracing calls are skipped or delayed, the resulting Graph of Trace can become incomplete or fragmented. As a result, trace quality currently depends on the agent's compliance with the instrumentation protocol.

Monitor Interpretation Errors. The framework uses an LLM-based interpretation module to convert low-level execution reports into higher-level subtask units. Although we apply structured prompts and output validation to improve reliability, this module can still misinterpret ambiguous reports or introduce hallucinated structure. Such errors may lead to incorrect task decomposition or inaccurate dependency links.

Scalability of Large Execution Graphs. For extremely long workflows that may generate hundreds of nodes, the current framework still lacks richer interaction mechanisms such as hierarchical subgraph collapsing, semantic module grouping, and progressive node expansion. This can increase the cognitive burden on users when inspecting long-term tasks.

References

Emily J. Allen, Ghislain St-Yves, Yihan Wu, Jesse L. Breedlove, Jacob S. Prince, Logan T. Dowdle, Matthias Nau, Brad Caron, Franco Pestilli, Ian Charest, J. Benjamin Hutchinson, Thomas Naselaris, and Kendrick Kay. 2022. A massive 7T fMRI dataset to bridge cognitive neuroscience and artificial intelligence. *Nature Neuroscience*, 25(1):116–126.

Filippos Bellos, Yayuan Li, Cary Shu, Ruey Day, Jeffrey Siskind, and Jason Corso. 2025. Towards effective human-in-the-loop assistive ai agents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2513–2522.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, and 1 others. 2024. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*.

Yuxiao Cheng and Jinli Suo. 2025. Openlens ai: Fully autonomous research agent for health infomatics. *Preprint*, arXiv:2509.14778.

Keyan Ding, Jing Yu, Junjie Huang, Yuchen Yang, Qiang Zhang, and Huajun Chen. 2025. SciToolAgent: a knowledge-graph-driven scientific agent for multitool integration. *Nature Computational Science*, 5(10):962–972.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J. Szostkiewicz, Jon M. Laurent, Muhammed T. Razzak, Andrew D. White, Michaela M. Hinks, and Samuel G. Rodriques. 2025. Robin: A multi-agent system for automating scientific discovery. *Preprint*, arXiv:2505.13400.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2026. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Trans. Softw. Eng. Methodol.*

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, Di Yin, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, and 4 others. 2025a. Biomni: A General-Purpose Biomedical AI Agent. *bioRxiv*, page 2025.05.30.656746.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, Kun Shao, and Jun Wang. 2025b. Deep research agents: A systematic examination and roadmap. *Preprint*, arXiv:2506.18096.

Zongqian Li, Ehsan Shareghi, and Nigel Collier. 2025. Reasongraph: Visualisation of reasoning paths.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *Preprint*, arXiv:2408.06292.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery.

Shuo Ren, Can Xie, Pu Jian, Zhenjiang Ren, Chunlin Leng, and Jiajun Zhang. 2026. Towards Scientific Intelligence: A Survey of LLM-based Scientific Agents. *arXiv preprint*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043.

Erzhuo Shao, Yifang Wang, Yifan Qian, Zhenyu Pan, Han Liu, and Dashun Wang. 2025. SciSciGPT: advancing human–AI collaboration in the science of science. *Nature Computational Science*.

Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, Jianing Wang, Qintong Li, Xiangru Tang, Tianbao Xie, Xiachong Feng, Xiang Li, Ben Kao, Wenhai Wang, Biqing Qi, and 2 others. 2025. ScienceBoard: Evaluating Multimodal Autonomous Agents in Realistic Scientific Workflows. *arXiv preprint*.

Michael Tangermann, Klaus-Robert Müller, Ad Aertsen, Niels Birbaumer, Christoph Braun, Clemens Brunner, Robert Leeb, Carsten Mehring, Kai J. Miller, Gernot R. Müller-Putz, Guido Nolte, Gert Pfurtscheller, Hubert Preissl, Gerwin Schalk, Alois Schlögl, Carmen Vidaurre, Stephan Waldert, and Benjamin Blankertz. 2012. Review of the BCI Competition IV. *Frontiers in Neuroscience*, 6.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. 2025. The openhands software agent sdk: A composable and extensible foundation for production agents.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Mohammad Yaghoubi, M Ganesh Kumar, Andres Nieto-Posadas, Coralie-Anne Mosser, Thomas Gisiger, Émmanuel Wilson, Cengiz Pehlevan, Sylvain Williams, and Mark P Brandon. 2026. Predictive coding of reward in the hippocampus. *Nature*, pages 1–7.

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

Weike Zhao, Chaoyi Wu, Yanjie Fan, Pengcheng Qiu, Xiaoman Zhang, Yuze Sun, Xiao Zhou, Shuju Zhang, Yu Peng, Yanfeng Wang, Xin Sun, Ya Zhang, Yongguo Yu, Kun Sun, and Weidi Xie. 2026. An agentic system for rare disease diagnosis with traceable reasoning. *Nature*.

Wei-Long Zheng and Bao-Liang Lu. 2015. Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks. *IEEE Transactions on Autonomous Mental Development*, 7(3):162–175.

A Appendix

A.1 Prompts

The following prompt is designed for the Monitor Agent, which is to receive execution information from the Scientific Agent and generate structured execution nodes. To ensure generation correctness, the prompt explicitly defines the schema and semantics of a node. In addition to node generation, a critical responsibility of the Monitor is to establish parent-child relationships between nodes. The Monitor should choose a valid, logically consistent, and existing node as the parent. We therefore define explicit rules in the prompt to constrain this process. To further improve reliability, we include representative examples in the prompt. In our experiments, these examples significantly improve the Monitor's performance.

Prompts: Monitor

```
You are a strict information
extractor. Convert the provided
subtask summary into one or more
DAG nodes.

<Node Definition>
A node represents a research-
relevant operation that
contributes to the experimental,
analytical, or infrastructural
progression of the research.

<Node Boundary and Splitting Rules>
Split actions into separate nodes
IF AND ONLY IF:
1. One action explicitly depends
on the result of another.
2. Each action has independent
research-level semantic
significance.
...
<Parent Selection Rules>
- Each node MUST have at least one
parent.
- Parents represent logical
research justification, NOT
chronological execution order.
- You MUST select parent ids from
existing nodes.
...
<Artifacts Requirements>
Each node MUST include an
artifacts list (may be empty
only if no concrete file artifact
exists).
Artifacts represent the verifiable
output of the node.
Rules:
- Execution, visualization,
analysis, and conclusion nodes
MUST produce concrete, inspectable
artifacts.
- Setup or literature nodes MAY
have empty artifacts if no file-
level output exists.

<Examples>
...
<Previous Graph of Trace>
...
<Output Schema>
...
```

We provide an MCP description to guide the Scientific Agent on how to call the Monitor Agent. To maximize information flow into the Monitor, we encourage the Scientific Agent to invoke the MCP after completing each subtask, ensuring that the Monitor receives better execution information, while avoiding unnecessary verbosity such as raw logs or debugging details. Additionally, we also include several representative examples for timely and properly formatted Monitor Agent calls.

Prompts: MCP Description

```
This tool records **what was
done** (an executable, verifiable
operation) and its artifact
dependencies.
It is NOT for chain-of-thought,
progress logs, explanations, or
exploratory work.

<Call Gate>
Each MCP call record MUST
represent exactly ONE executable
action step.
Do NOT call for planning,
```

```
debugging, or rephrasing.
After completing any subtask, you
must immediately record it via the
MCP tool.
When in doubt about whether the
step qualifies or not, you must
still record it.
...
<Input Requirements>
Provide exactly ONE subtask
describing a minimal executable
unit with a factual `title` and
`description`.
If this subtask depends on
prior results/steps, provide a
dependence list to describe those
dependencies.
Dependence list MAY include
multiple items.
...

<EXAMPLES>
...
```

B Expert Evaluation Questionnaire

We design a comprehensive questionnaire and conduct a structured expert evaluation of the visualization framework, covering structural validity, node and edge level fidelity, latency, and usability. The results are reported in Table 1.

B.1 Structural Validity

The times of failed rendering during execution: ___

B.2 Node-Level Fidelity

Total nodes: ___ Accurate nodes: ___

B.3 Relation-Level Fidelity

Total edges: ___ Correct edges: ___

B.4 Temporal Consistency

Update latency:
☐ Real-time (<2s) ☐ Slight (2–5s)
☐ Moderate (5–10s) ☐ Severe (10–20s)
☐ Unacceptable (>20s / unstable)

B.5 Expert Interpretability

☐ Immediate Understanding
☐ Minor clarification needed
☐ Partially with effort needed
☐ Fragmented
☐ No effective understanding

B.6 Visual Aesthetics

☐ Excellent ☐ Good ☐ Fair ☐ Poor
☐ Very poor

B.7 Usability

☐ Highly convenient
☐ Convenient
☐ Moderately convenient
☐ Inconvenient
☐ Very inconvenient