# TinyScientist: An Interactive, Extensible, and Controllable Framework for Building Research Agents

**Haofei Yu**[1*] **Keyang Xuan**[1*] **Fenghai Li**[1*]
**Kunlun Zhu**[1] **Zijie Lei**[1] **Jiaxun Zhang**[1] **Ziheng Qi**[1]
**Kyle Richardson**[2] **Jiaxuan You**[1]
[1]University of Illinois Urbana-Champaign,
[2]Allen Institute for Artificial Intelligence

## Abstract

Automatic research with Large Language Models (LLMs) is rapidly gaining importance, driving the development of increasingly complex workflows involving multi-agent systems, planning, tool usage, code execution, and human-agent interaction to accelerate research processes. However, as more researchers and developers begin to use and build upon these tools and platforms, the complexity and difficulty of extending and maintaining such agentic workflows have become a significant challenge, particularly as algorithms and architectures continue to advance. To address this growing complexity, TinyScientist identifies the essential components of the automatic research workflow and proposes an interactive, extensible, and controllable framework that adapts easily to new tools and supports iterative growth. We provide an open-source codebase[1], an interactive web demonstration[2], and a PyPI Python package[3] to make state-of-the-art auto-research pipelines broadly accessible to every researcher and developer.

## 1 Introduction

Interest in building research agents with Large Language Models (LLMs) to interact with human researchers and enable automatic scientific discovery has gained considerable attention in recent years (Gottweis et al., 2025). Such agentic frameworks have demonstrated impressive capabilities across a wide range of research tasks, including ideation (Si et al., 2024; Li et al., 2024a), scientific coding (Chan et al., 2024; Huang et al., 2023), paper writing (Wang et al., 2024), review writing (Jin et al., 2024), and even end-to-end research pipelines (Jansen et al., 2025; Lu et al., 2024; Yamada et al., 2025; Li et al., 2024b; Cheng et al., 2025). Recent advances in this area leverage methods including multi-agent collaboration (Schmidgall et al., 2025), tool using (Skarlinski et al., 2024), and tree-based search (Yamada et al., 2025) to augment its performance.

In spite of this success, however, existing automatic research systems often design and use agentic frameworks that are overly complex and difficult to use and extend without significant technical expertise. These challenges stem from three key issues: (1) ***lack of interactivity***: human researchers struggle to engage with the specific agent's research progress due to the complexity of research intents and unclear communication interfaces (Zou et al., 2025; Liu et al., 2025b), making feedback incorporation challenging. (2) ***limited extensibility***: existing representative frameworks rely on rigid, tool-specific designs (Zhang et al., 2025), making it hard to integrate new tools or adapt to different research domains. (3) ***insufficient controllability***: many systems offer weak supervision on safety, ethical constraints, and cost budget, raising concerns about misalignment and unbounded execution (Gridach et al., 2025; Liu et al., 2025a). To address these issues and help democratize the development and use of research agents, we introduce TinyScientist, a lightweight and modular agentic framework that facilitates interactivity, extensibility, and controllability, making it highly accessible to users, researchers, and developers. Specifically, TinyScientist is designed based on the following principles illustrated in Figure 1.

**Interactivity**. Research is an open-ended process that requires continuous user involvement. Researchers typically begin with vague or evolving goals and refine them over time. As a result, effective research agents must support real-time adjustments to their reasoning, coding, and writing

*Core Contributors.

[1]The codebase for TinyScientist is available at `https://github.com/ulab-uiuc/tiny-scientist`.

[2]The web demonstration for TinyScientist is hosted at `https://app.auto-research.dev`.

[3]The Python package for TinyScientist is released at `https://pypi.org/project/tiny-scientist`.

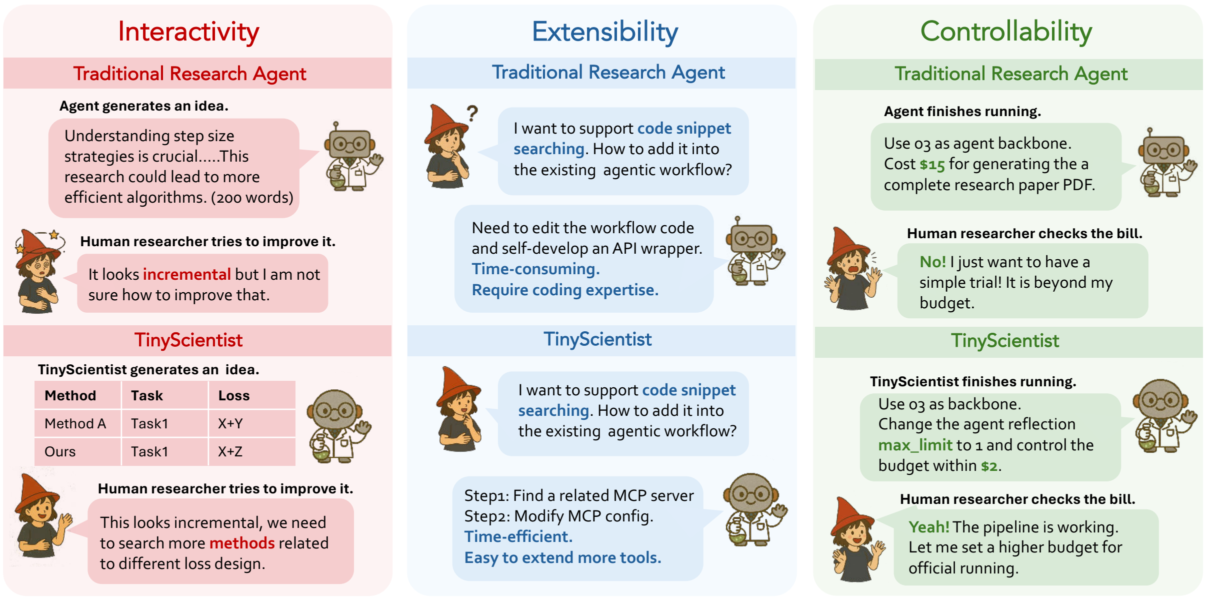

Interactivity
Traditional Research Agent
Agent generates an idea.
Understanding step size strategies is crucial.....This research could lead to more efficient algorithms. (200 words)
Human researcher tries to improve it.
It looks incremental but I am not sure how to improve that.
TinyScientist
TinyScientist generates an idea.
Method | Task | Loss
Method A | Task1 | X+Y
Ours | Task1 | X+Z
Human researcher tries to improve it.
This looks incremental, we need to search more methods related to different loss design.
Extensibility
Traditional Research Agent
I want to support code snippet searching. How to add it into the existing agentic workflow?
Need to edit the workflow code and self-develop an API wrapper. Time-consuming. Require coding expertise.
TinyScientist
I want to support code snippet searching. How to add it into the existing agentic workflow?
Step1: Find a related MCP server Step2: Modify MCP config. Time-efficient. Easy to extend more tools.
Controllability
Traditional Research Agent
Agent finishes running.
Use o3 as agent backbone. Cost $15 for generating the a complete research paper PDF.
Human researcher checks the bill.
No! I just want to have a simple trial! It is beyond my budget.
TinyScientist
TinyScientist finishes running.
Use o3 as backbone. Change the agent reflection max_limit to 1 and control the budget within $2.
Human researcher checks the bill.
Yeah! The pipeline is working. Let me set a higher budget for official running.

Figure 1: **Design principles for TINYSCIENTIST.** We highlight the key differences between traditional research agents and TINYSCIENTIST. To enhance *interactivity*, TINYSCIENTIST introduces a table-based interface that helps researchers clearly express and refine their intents. For *extensibility*, TINYSCIENTIST adopts an MCP (Model Context Protocol) design instead of direct API wrapping, making it easy to add or replace tools. For *controllability*, TINYSCIENTIST includes built-in safety and budget controllers that monitor and regulate the entire workflow.

processes. Without an interactive interface, agentic frameworks risk drifting from the user's intent and producing irrelevant or unsafe outputs. To address this, TINYSCIENTIST introduces a modular, tabular-based interface that decomposes the research workflow into editable stages. Each stage presents intermediate results in a structured tabular format, allowing researchers to directly modify specific cells or columns—*e.g.*, by suggesting adding new baselines as rows or editing individual entries. Such a design improves clarity in human-agent communication and empowers users to guide the system as their objectives evolve.

**Extensibility**. Automatic research is evolving rapidly, with new tools and technologies emerging constantly. To keep pace, agentic frameworks need to support the easy integration and replacement of tools. In machine learning-related automatic research, while the core workflow—typically composed of stages like think, code, write, and review—remains relatively fixed, the key difference between different tasks lies in the tools and methods used within each stage. To address this, TINYSCIENTIST adopts the design of the Model Context Protocol (MCP) (Anthropic, 2024), which provides a unified API for connecting diverse tools to augment each core workflow component. Such a system architecture enables seamless extension, allowing developers to upgrade and maintain their system with little effort.

**Controllability**. Safety, ethical, and financial concerns are critical, but often under-addressed in agentic research workflows (Zhu et al., 2025a; Tang et al., 2024). Users should not be caught off guard by excessive spending or unsafe outputs. Users should not be exposed to unexpected costs, unsafe behaviors, or misaligned actions. To ensure responsible and predictable execution, TINYSCIENTIST emphasizes controllability across the entire pipeline. Users are allowed to set explicit upper bounds on key hyperparameters—such as the number of experimental runs or self-reflection steps—to ensure budget constraints are respected. Additionally, at each stage of the workflow, built-in safety checkers validate outputs to prevent harmful or unintended behavior, which helps to maintain alignment with user intents.

To demonstrate the effectiveness of TINYSCIENTIST, we conduct both qualitative and quantitative evaluations, highlighting four key advantages: (1) It is easy for researchers to use the Python package without configuration or setup barriers. (2) It enables better human–agent interaction through an interactive UI. (3) It achieves research generation quality comparable to Agent Laboratory (Schmidgall et al., 2025), a widely used multi-agent auto-research framework, and (4) using tools improves the generation quality.

| Framework | Interactivity | | Extensibility | | Controllability | | Deployment | |
|---|---|---|---|---|---|---|---|---|
| | Modular Design | Tabular Commun. | Tool Calling | Schema Diagram | Safety Control | Budget Control | UI Design | Python Package |
| AI Scientist (Lu et al., 2024) | ✓ | ✗ | API | ✗ | ✓ | ✗ | ✗ | ✗ |
| AI co-scientist (Gottweis et al., 2025) | ✓ | ✗ | API | ✗ | ✗ | ✗ | ✓ | ✗ |
| AI Researcher (Tang et al., 2025) | ✓ | ✗ | API | ✗ | ✗ | ✗ | ✗ | ✗ |
| Agent Laboratory (Schmidgall et al., 2025) | ✓ | ✗ | wrapper | ✗ | ✓ | ✗ | ✗ | ✗ |
| **TinyScientist (ours)** | ✓ | ✓ | MCP | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 1: **Comparison of research agent frameworks.** We compare existing frameworks across four key dimensions: interactivity, controllability, extensibility, and deployment readiness. TINYSCIENTIST uniquely integrates tabular-based human-agent communication, schematic diagram design, MCP-based tool calling, and budget/safety control, all within a deployment-ready system. For tool calling, *API* refers to frameworks that directly invoke APIs as part of their workflow without abstraction. In contrast, *wrapper* denotes systems that support tool abstraction and allow users to wrap custom tools via APIs, though without a standardized integration method like MCP.

## 2 Related Work

**Agentic workflow for automatic research**. The field of automatic research has witnessed rapid advancements in recent years, with diverse frameworks emerging to automate the research process through various design principles and coordination strategies. For example, AI-Scientist (Lu et al., 2024) introduced the first comprehensive fully-automatic research agent by enabling frontier LLMs to conduct a series of research processes. In subsequent work, AI-Scientist v2 (Yamada et al., 2025) improves the pipeline by replacing manual templates with an agentic tree-search methodology and a VLM-based feedback loop. In addition, later work such as Agent Laboratory (Schmidgall et al., 2025) and AI-Researcher (Tang et al., 2025) follow a similar staged design to develop end-to-end autonomous research workflows, introducing more refined role specialization and enhanced coordination mechanisms. Furthermore, AI co-scientist (Gottweis et al., 2025) and ResearchTown (Yu et al., 2024) utilize a specialized multi-agent coordination paradigm to facilitate novel scientific idea discovery. While prior work pursues automation through complex orchestration, our work prioritizes simplicity and modularity by distilling the research process into four core stages, striking a balance between automation, simplicity, and extensibility.

**Human-in-the-loop for automatic research**. While fully automated research pipelines are promising, they remain practically limited without human involvement, underscoring the need for human oversight. Recognizing this, recent work has incorporated human-in-the-loop functionality that allows researchers to contribute to different stages of automated research. For the idea stage, Garikaparthi et al. (2025) and Radensky et al. (2024) support interactive hypothesis refinement and facet recombination with researcher feedback. In addition, CodeScientist (Jansen et al., 2025) introduces an end-to-end system for semi-automated scientific discovery where humans can collaborate with LLMs to design, execute, and interpret code-based scientific experiments. Furthermore, Ifargan et al. (2025) and DeepReview (Zhu et al., 2025b) incorporate human experts' feedback to review and refine LLM-generated scientific drafts, ensuring alignment with expert judgment. Our work follows this trend by treating human feedback as the central component, particularly through our table-based user interface design.

## 3 TINYSCIENTIST Framework

In this section, we first provide a brief overview of TINYSCIENTIST. We then describe each core module in the agentic workflow backbone. Finally, we introduce the features built on top of this workflow that make TINYSCIENTIST interactive, extensible, and controllable.

### 3.1 Framework Overview

The ultimate goal behind TINYSCIENTIST is to minimize the complexity of research agent workflows and make them accessible to everyone. To achieve this, we first clarify the input/output design of our framework. We then describe the hierarchical and modularized component architecture included in our framework.

**Framework I/O**. To support general use, TINYSCIENTIST accommodates multiple input and output formats. We identify three common types of re-

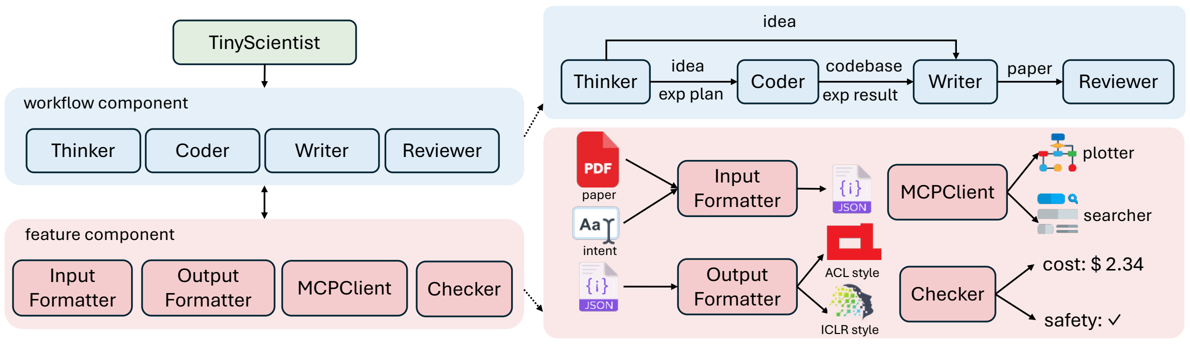


Figure 2: **Overview of TinyScientist framework**. On the left side, the diagram illustrates the class hierarchy of TinyScientist. At the top-left side, `TinyScientist` serves as the base class. It manages four workflow components, each responsible for a core stage of the research process. In turn, each workflow component is supported by four feature components that enhance its functionality beyond its core function. On the right side, the overall workflow and the details for each feature component are described separately.

search data: AI conference-style PDFs (*e.g.*, ACL[4] and ICLR[5]), structured JSON data, and plain text strings. The I/O design of TinyScientist accepts any of these formats as input and can generate output in any of them. For example, a typical use case is taking a plain-text intent as input and producing a fully formatted conference paper PDF as output.

**Framework architecture**. As shown in Figure 2, our framework is organized into a clear, hierarchical architecture. At the top, the Engine class orchestrates four core *workflow components* (thinker, coder, writer, and reviewer), and each represents a distinct stage in the research lifecycle. These components are modularized for interactivity, allowing users to inspect and guide the agent at each step. Each workflow component is further supported by a set of reusable *feature components*, including InputFormatter, OutputFormatter, MCPClient, and Checker. These components are designed with specific goals in mind: extensibility through MCPClient for flexible tool integration, and controllability through the Checker for enforcing safety and budget constraints. Together, the separation between workflow and feature components ensures a clean framework architecture.

### 3.2 Workflow Components

In this section, we first describe the overall agentic structure and the specific functionality of each workflow component.

**Basic: Iterative agent**. Each agent follows an iterative, self-refinement paradigm (Renze and Guven, 2024; Madaan et al., 2023), where it repeatedly performs and improves upon a task until reaching a predefined iteration cap. This shared iterative structure applies to all stages in the workflow.

**Stage1: Think**. The *Thinker* module is responsible for research ideation based on the user's input intent. It samples $n$ initial ideas via LLM-based prompting, where each idea is refined through $k$ rounds of iterative improvement. Each idea contains: (1) a descriptive paragraph; (2) an experimental plan; (3) a comparison table with related works. Additionally, the Thinker provides self-evaluation scores for each idea along three dimensions: impact, feasibility, and novelty. Formally, we express this process as the following transformation:

$$\texttt{Thinker(intent)} \rightarrow \texttt{idea}$$

**Stage2: Code**. Given an idea and its experimental plan, the *Coder* module leverages an external coding agent framework (*e.g.*, `Aider`[6]) to iteratively generate executable codebase and run experiments. If the execution fails or deviates from the plan, the agent pauses and awaits human input. Conversely, upon successful execution, it returns the experimental results and associated code artifacts to the output directory. Abstractly, this takes the following form:

$$\texttt{Coder(idea)} \rightarrow \texttt{codebase}$$

**Stage3: Write**. The *Writer* module treats scientific paper writing as a structured three-step process: (1) initial generation, (2) paper refinement, and (3) citation insertion. For each paper section (*e.g.*, Introduction), the writer receives structured inputs, such

[4]We refer to using the ACL conference template as `https://github.com/acl-org/acl-style-files`

[5]We refer to using the ICLR conference template as `https://github.com/ICLR/Master-Template`

[6]We refer to `https://github.com/Aider-AI/aider`

as an idea and a codebase, then generates a draft using an LLM. The draft is then refined based on an error checklist to fix LaTeX format inconsistencies. Finally, citation embedding is performed by retrieving relevant references $\{r_1, r_2, \ldots, r_n\}$ via the Semantic Scholar API (Kinney et al., 2023)[7], and inserting them into the completed draft to yield the final version. As above, we can define this as:

$$\texttt{Writer(idea, codebase)} \rightarrow \texttt{paper}$$

**Stage4: Review**. The *Reviewer* module evaluates the completed paper and simulates peer reviews, each including a summary, strengths, and weaknesses in standard format. Every review is refined through self-reflection, and a meta-review is synthesized along with a final score:

$$\texttt{Reviewer(paper)} \rightarrow \texttt{review}$$

### 3.3 Feature Components

Beyond the four workflow modules discussed in Section §3.2, TinyScientist introduces three key feature components—Formatter, MCPClient, and Checker—to support its core principles: interactivity, extensibility, and controllability, respectively.

**Formatter: Enhancing interactivity via tabular-based communication**. In automatic research, agents often generate large amounts of intermediate information, making it difficult for human researchers to track progress and monitor individual steps. To address this, TinyScientist uses the Formatter to compile key outputs into structured tables between different workflow components. These tables provide a clear summary and comparison of the agent's progress, enabling researchers to easily review, comment on, and directly edit specific elements, thereby facilitating precise and interactive guidance throughout the workflow. Figure 4 shows a concrete example of the table generated by LLMs for research ideation.

**MCPClient: Enhancing extensibility via tool integration**. Modern research workflows require support beyond LLM prompting. MCPClient serves as a bridge between workflow components and a wide range of research tools—such as code searchers, plot drawers, and paper retrievers—enabling seamless integration and future extensibility. Appendix §D provides an example of how MCP is used to enhance paper writing by incorporating schematic diagram generation.

[7]We refer to `https://api.semanticscholar.org/`

**Checker: Enhancing controllability via budget and safety constraints**. To ensure controllable usage, the Checker module enforces constraints on cost and safety. Users can set limits (*e.g.*, model size, max iterations), and the system adjusts parameters like the number of self-reflections accordingly. Stage-specific safety filters (*Thinker*, *Coder*, *Reviewer*) proactively block harmful outputs. Table 21 shows an example for the safety checker.

## 4 TinyScientist Python Package

To demonstrate the practicality of TinyScientist, we develop a Python package based on the proposed agentic framework, enabling easy and modular use. As illustrated in Algorithm 1, with just seven lines of code, the package can generate a fully compiled PDF of a complete research paper in the standard AI conference format - along with the research idea, the corresponding experimental code, and details of the peer review process.

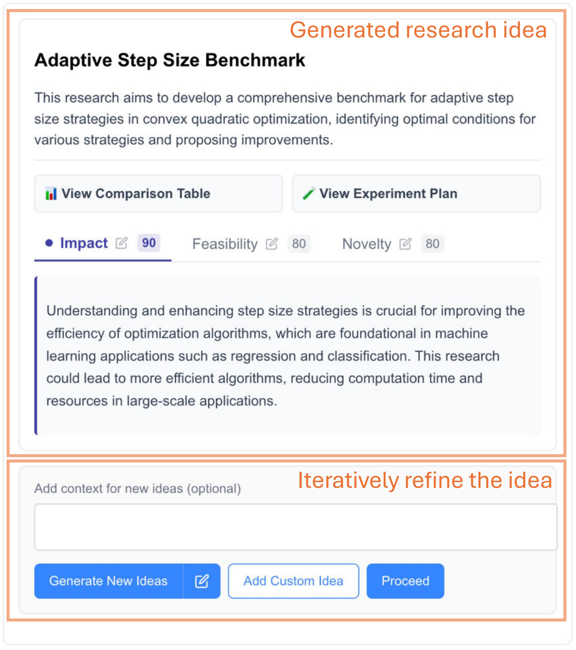
**Algorithm 1** TinyScientist usage example

```
# model: string name for LLM
# intent: string of user intent description

from tiny_scientist import TinyScientist
# Instantiate TinyScientist
scientist = TinyScientist(model)
# Idea Generation
idea = scientist.think(intent)
# Code Experiment
status, exp_dir = scientist.code(idea)
if status:
  # Paper Writing
  pdf_path = scientist.write(idea, exp_dir)
  # Paper Review
  review = scientist.review(pdf_path)
```

## 5 TinyScientist User Interface

To further enhance the usability of TinyScientist, we develop a user interface that leverages the TinyScientist Python package to build its backend. The interactive and modular nature of the TinyScientist design lends itself to an intuitive UI design. In the UI, each workflow stage is presented on a dedicated page, arranged sequentially. Details about the UI design are available at Appendix §A.

**Iterative interaction within one stage**. This feature arises from the iterative agent design introduced in Section §3.2. Since each core workflow component supports iterative refinement, TinyScientist allows fine-grained user inputs to be injected during the agent's reflection process. As

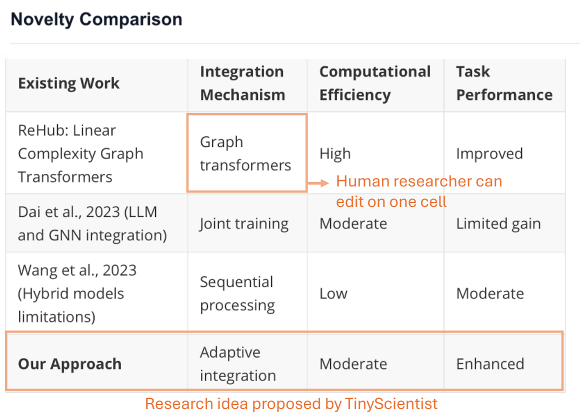


**Adaptive Step Size Benchmark**

This research aims to develop a comprehensive benchmark for adaptive step size strategies in convex quadratic optimization, identifying optimal conditions for various strategies and proposing improvements.

Understanding and enhancing step size strategies is crucial for improving the efficiency of optimization algorithms, which are foundational in machine learning applications such as regression and classification. This research could lead to more efficient algorithms, reducing computation time and resources in large-scale applications.

Figure 3: **Example of iterative interaction within the thinking stage.** The upper box shows a research idea (including contents, scores, tables, and experimental plans) generated by the Thinker. The lower box allows users to provide custom instructions to refine.


Human researcher can edit on one cell
Research idea proposed by TinyScientist

**Novelty Comparison**

| Existing Work | Integration Mechanism | Computational Efficiency | Task Performance |
|---|---|---|---|
| ReHub: Linear Complexity Graph Transformers | Graph transformers | High | Improved |
| Dai et al., 2023 (LLM and GNN integration) | Joint training | Moderate | Limited gain |
| Wang et al., 2023 (Hybrid models limitations) | Sequential processing | Low | Moderate |
| **Our Approach** | Adaptive integration | Moderate | Enhanced |

Figure 4: **Example for tabular-based interaction between stages**. This shows one novelty comparison result of idea thinking. It organized the generated idea as one line within one table.

shown in Figure 3, human researchers can continuously add or adjust intents, prompting the system to refine and regenerate ideas accordingly.

**Tabular-based interaction across stages**. This functionality is built upon the tabular-based communication mechanism discussed in Section §3.3. In TinyScientist, generated ideas and experimental plans are organized into structured tables, making them easy to interpret and edit. Figure 4 presents an example of a table generated by TinyScientist for novelty comparison. The tabular format highlights key differences between generated ideas and prior work, allowing human researchers to clearly understand, compare, and modify content as needed.

## 6 Evaluation Results

In addition to qualitatively analyzing the Python package and user interface of TinyScientist, we conduct quantitative evaluations to verify that our agentic framework—designed for enhanced interactivity, extensibility, and controllability—maintains the quality of generated papers.

### 6.1 Evaluation Settings

**Model settings**. We use `gpt-4o-mini` as the backbone model for both TinyScientist and Agent Laboratory to ensure a fair comparison between the two agentic frameworks.

**Data settings**. We evaluate two categories of tasks. (1) *In-distribution tasks:* We randomly sample 20 machine learning-related ideas from Si et al. (2024), using their titles as user inputs. (2) *Out-of-distribution tasks:* To test the safety and robustness of TinyScientist, we randomly sample 20 biology-related potentially unsafe tasks (*e.g.*, DNA synthesis for synthetic genomes) from SciSafetyBench (Zhu et al., 2025a) and use them as input intents for both frameworks.

**Evaluation settings**. We conduct both automated and human evaluations. For the automated evaluation, we use a multi-agent LLM-based pipeline in which three LLMs provide independent reviews, and a meta-review aggregates them into a final judgment. For the human evaluation, we follow the rubrics in Appendix §C, with annotators who have relevant research backgrounds assessing the quality of the generated papers from both frameworks.

### 6.2 Evaluation Results

**TinyScientist achieves comparable generation quality to Agent Laboratory**. As shown in Figure 5, in our in-distribution tasks, human evaluation rates TinyScientist about 0.5 points higher on average than Agent Laboratory, while LLM-based evaluation gives Agent Laboratory a slight advantage. Human annotators observes that papers produced by Agent Laboratory tended to include fewer novel and more similar ideas compared to those from TinyScientist. In addition, papers generated by TinyScientist often contained tables and figures that improved readability and com-

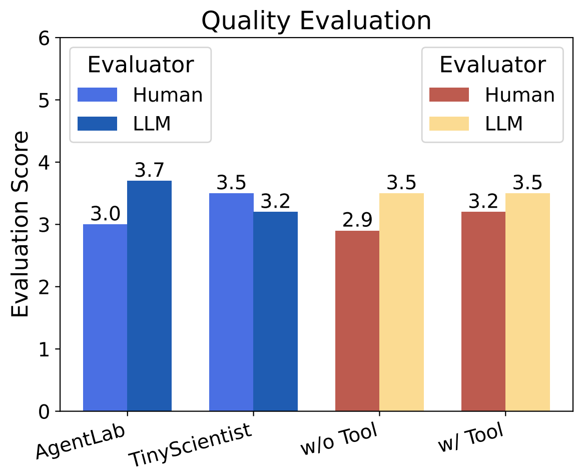


Figure 5: **Quality evaluation results**. We report both human and LLM-based quality scores (1–5) for generated paper outputs. The left side compares paper quality between Agent Laboratory and TinyScientist. The right side conducts an ablation study within TinyScientist, evaluating the effect of tool usage.

prehension for human researchers. Overall, we find that TinyScientist achieves a quality comparable to Agent Laboratory, while offering greater user-friendliness and lower cost.

**Tool use of TinyScientist provides a slight improvement in the paper quality**. We further conduct a focused evaluation on the effect of tool usage within TinyScientist for in-distribution tasks. As shown on the right side of Figure 5, human ratings indicate a modest improvement in quality when tools are employed. This improvement is likely due to the use of information-retrieval tools (*e.g.*, searchers), which enrich the generation with more relevant and diverse content.

**TinyScientist blocks unsafe user intents effectively**. To evaluate the controllability of TinyScientist, we conduct evaluations under out-of-distribution tasks in biological domains. Among the total 20 tasks, 18 tasks were blocked at the thinker stage, while the remaining 2 were flagged with warnings at the further workflow stages. These results demonstrate that the checker enables TinyScientist to effectively prevent the development of potentially harmful research.

## 7 Conclusion

In this work, we present TinyScientist, a lightweight agentic framework that prioritizes simplicity and usability to democratize the development of research agents. To enhance accessibility, we developed an easy-to-use Python package and a highly interactive web demonstration for this framework. We believe that TinyScientist will help lower the barrier for both researchers and developers to enter the field of automatic research, encouraging more researchers to adopt and contribute to research agent development.

## Ethical Statements and Broader Impacts

The development of TinyScientist targets for extensive and interactive Automated research workflows carries inherent ethical risks, including the potential for misuse, unintended harm, or malicious exploitation. To mitigate these concerns, TinyScientist incorporates multiple safeguards: (1) configurable budget and safety controls, (2) automated watermarking on generated papers to clearly indicate AI involvement, and (3) an interactive user interface that supports real-time human oversight and intervention. We emphasize that TinyScientist is designed to augment—not replace—human researchers. Its goal is to accelerate scientific discovery through transparent, collaborative human-AI workflows, not to enable fully autonomous research without accountability. To further uphold ethical standards, we openly release TinyScientist as a Python library with user-friendly interfaces, making advanced research capabilities accessible to a broader community while maintaining transparency and control.

Regarding generated data, we acknowledge that some AI-generated artifacts (*e.g.*, papers or figures) may closely resemble human-written content. To prevent misuse, all generated outputs are clearly watermarked and not intended for direct use in academic publishing without human verification. This aligns with best practices for responsible research and ensures that the system and its outputs are used in ethically appropriate ways.

## Limitations

There are two main limitations for our work:

**Compilation stability**. While our system performs robustly in most scenarios, compilation failures occasionally arise due to inconsistencies in LaTeX formatting, which causes issues such as references missing or line misalignment. Improved context sanitizer and format-control generation are needed to ensure stability across all outputs.

**Diagram quality and informativeness**. Although our system can generate diagrams for key sections such as the Introduction and Method, the visual quality and informativeness of these figures remain limited. The generated SVGs often lack precise alignment, which reduces their effectiveness in conveying the core information of the paper. Improving visual consistency and content-grounding in diagram generation would significantly enhance the entire paper's clarity.

## Acknowledgments

We sincerely appreciate the support from the Amazon grant funding project #120359, "GRAG: Enhance RAG Applications with Graph-structured Knowledge", and Meta gift funding project "PERM: Toward Parameter Efficient Foundation Models for Recommenders".

## References

Anthropic. 2024. Model context protocol. Accessed: 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, and 1 others. 2024. Mle-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*.

Junyan Cheng, Peter Clark, and Kyle Richardson. 2025. Language Modeling by Language Models. *arXiv preprint arXiv:2506.20249*.

Aniketh Garikaparthi, Manasi Patwardhan, Lovekesh Vig, and Arman Cohan. 2025. Iris: Interactive research ideation system for accelerating scientific discovery. *arXiv preprint arXiv:2504.16728*.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, and 1 others. 2025. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*.

Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. 2025. Agentic ai for scientific discovery: A survey of progress, challenges, and future directions. *arXiv preprint arXiv:2503.08979*.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2023. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*.

Tal Ifargan, Lukas Hafner, Maor Kern, Ori Alcalay, and Roy Kishony. 2025. Autonomous llm-driven research—from data to human-verifiable research papers. *NEJM AI*, 2(1):AIoa2400555.

Peter Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Daniel S Weld, and Peter Clark. 2025. Codescientist: End-to-end semi-automated scientific discovery with code-based experimentation. *arXiv preprint arXiv:2503.22708*.

Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. 2024. Agentreview: Exploring peer review dynamics with llm agents. *arXiv preprint arXiv:2406.12708*.

Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, and 1 others. 2023. The semantic scholar open data platform. *arXiv preprint arXiv:2301.10140*.

Ruochen Li, Liqiang Jing, Chi Han, Jiawei Zhou, and Xinya Du. 2024a. Learning to generate research idea with dynamic control. *arXiv preprint arXiv:2412.14626*.

Ruochen Li, Teerth Patel, Qingyun Wang, and Xinya Du. 2024b. Mlr-copilot: Autonomous machine learning research based on large language models agents.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, and 1 others. 2025a. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*.

Weiwen Liu, Jiarui Qin, Xu Huang, Xingshan Zeng, Yunjia Xi, Jianghao Lin, Chuhan Wu, Yasheng Wang, Lifeng Shang, Ruiming Tang, and 1 others. 2025b. The real barrier to llm agent usability is agentic roi. *arXiv preprint arXiv:2505.17767*.

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. *arXiv preprint arXiv:2408.06292*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Marissa Radensky, Simra Shahid, Raymond Fok, Pao Siangliulue, Tom Hope, and Daniel S Weld. 2024. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634*.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*.

Michael D Skarlinski, Sam Cox, Jon M Laurent, James D Braza, Michaela Hinks, Michael J Hammerling, Manvitha Ponnapati, Samuel G Rodriques, and Andrew D White. 2024. Language agents achieve superhuman synthesis of scientific knowledge. *arXiv preprint arXiv:2409.13740*.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2025. Ai-researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*.

Xiangru Tang, Qiao Jin, Kunlun Zhu, Tongxin Yuan, Yichi Zhang, Wangchunshu Zhou, Meng Qu, Yilun Zhao, Jian Tang, Zhuosheng Zhang, Arman Cohan, Zhiyong Lu, and Mark Gerstein. 2024. Prioritizing safeguarding over autonomy: Risks of llm agents for science.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Qingsong Wen, Wei Ye, and 1 others. 2024. Autosurvey: Large language models can automatically write surveys. *Advances in neural information processing systems*, 37:115119–115145.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

Haofei Yu, Zhaochen Hong, Zirui Cheng, Kunlun Zhu, Keyang Xuan, Jinwei Yao, Tao Feng, and Jiaxuan You. 2024. Researchtown: Simulator of human research community. *arXiv preprint arXiv:2412.17767*.

Wentao Zhang, Ce Cui, Yilei Zhao, Yang Liu, and Bo An. 2025. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv preprint arXiv:2506.12508*.

Kunlun Zhu, Jiaxun Zhang, Ziheng Qi, Nuoxing Shang, Zijia Liu, Peixuan Han, Yue Su, Haofei Yu, and Jiaxuan You. 2025a. Safescientist: Toward risk-aware scientific discoveries by llm agents.

Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. 2025b. Deepreview: Improving llm-based paper review with human-like deep thinking process. *arXiv preprint arXiv:2503.08569*.

Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, Hoang Nguyen, Yue Zhou, Weizhi Zhang, Liancheng Fang, Langzhou He, and 1 others. 2025. A survey on large language model based human-agent systems. *arXiv preprint arXiv:2505.00753*.

## A User Interface Details

In this section, we provide a step-by-step guide to the TINYSCIENTIST user interface, which consists of five main pages. The first is the *configuration input page* (Figure 6), where users provide API keys and select backbone models. After configuration, users proceed to the *intent input page* (Figure 7) to describe their research intent. Next, the *idea viewing page* (Figure 8) presents tree-structured idea generation results, together with tabular descriptions for novelty comparison and experiment plans. On this page, it also allows users to iteratively refine ideas by giving text-based feedback. Once an idea is confirmed, the interface moves to the *code viewing page* (Figure 9), where users can download the generated file structure. Finally, the *paper viewing page* (Figure 10) provides a PDF preview of the generated paper along with reviews generated from the review component of TINYSCIENTIST.

## B Prompting Details

In this section, we provide the details about the prompt used for each workflow stage in TINYSCIENTIST.

### B.1 Thinker prompt

We present the complete set of prompts used in the TINYSCIENTIST thinker module. The system prompt establishes the agent role as a research scientist and defines core guidelines for idea generation (Table 2). The idea generation and refinement prompts handle the creation and modification of research ideas through structured approaches to problem identification and solution development (Tables 3 and 4). The evaluation prompts provide a comprehensive assessment mechanism for evaluating research ideas (Table 5 and 6).

### B.2 Coder prompt

We present the complete set of prompts used in the TINYSCIENTIST coder module. The system prompt establishes the agent role as a research scientist and defines core guidelines for experiment implementation (Table 7). The code execution and error handling prompts handle the refinement and re-plan for the experiment in different scenarios (Tables 8), and the experiment format prompt controls the output format of essential experiment details (Tables 9).

### B.3 Writer prompt

We present the complete set of prompts used in the TINYSCIENTIST writer module. The system prompt establishes the agent role as a research scientist and defines core guidelines for each step of paper writing (Table 10). Section instructions prompts provide details, tips, and instructions for section writing (Table 11, Table 12). The citation management prompts are responsible for citation fetching and embedding (Table 13, Table 14, Table 15). If there is error in rendering PDF, we utilize the refinement prompt to solve (Table 16).

### B.4 Reviewer prompt

We present the complete set of prompts used in the TINYSCIENTIST reviewer module. The system prompt establishes the agent role as a research scientist and defines core guidelines for paper review (Table 17). Prompts Review Format & Refinement are responsible for providing detail review guidelines and format control (Table 18, Table 19).

## C Human Evaluation Details

We provide the detailed quality evaluation rubrics for human evaluation in Table 20.

## D Case Study

We first present a case study of the checker as part of the feature component, with its output shown in Table 21. For MCPClient, we provide a case study of the drawer, a commonly useful tool. The corresponding input is shown in Table 22, and the generated output is illustrated in Figure 11.

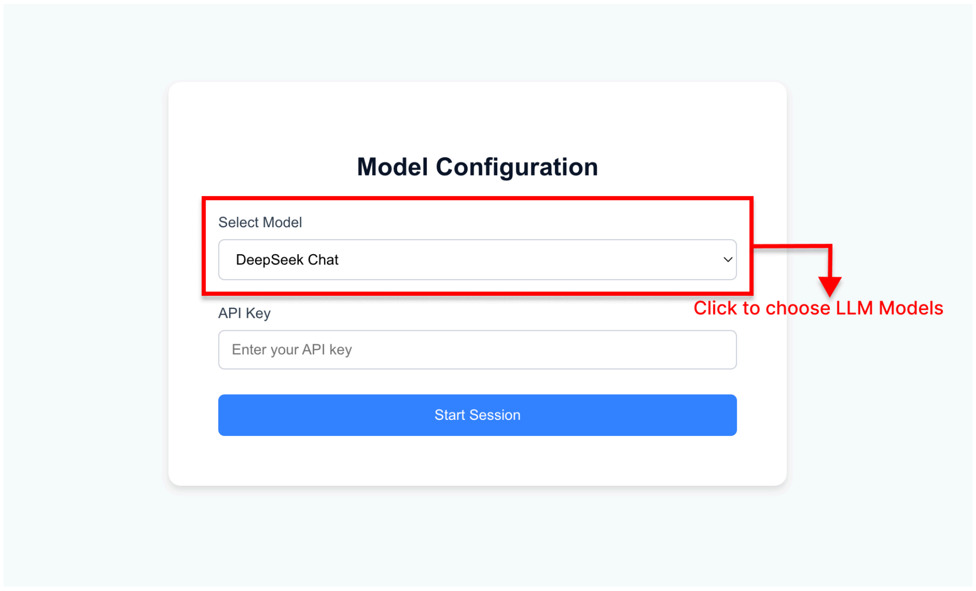
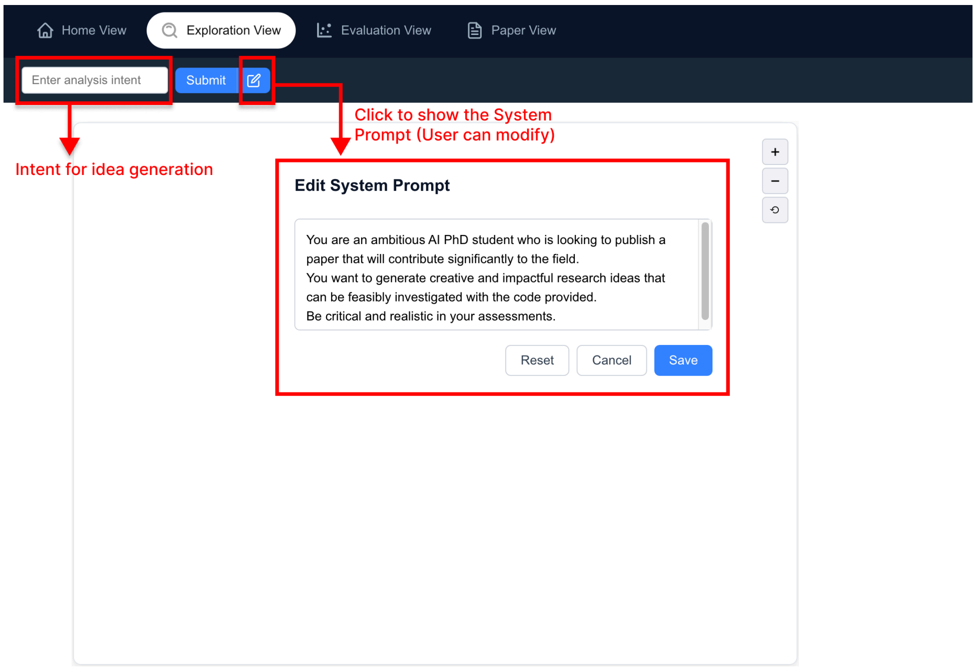
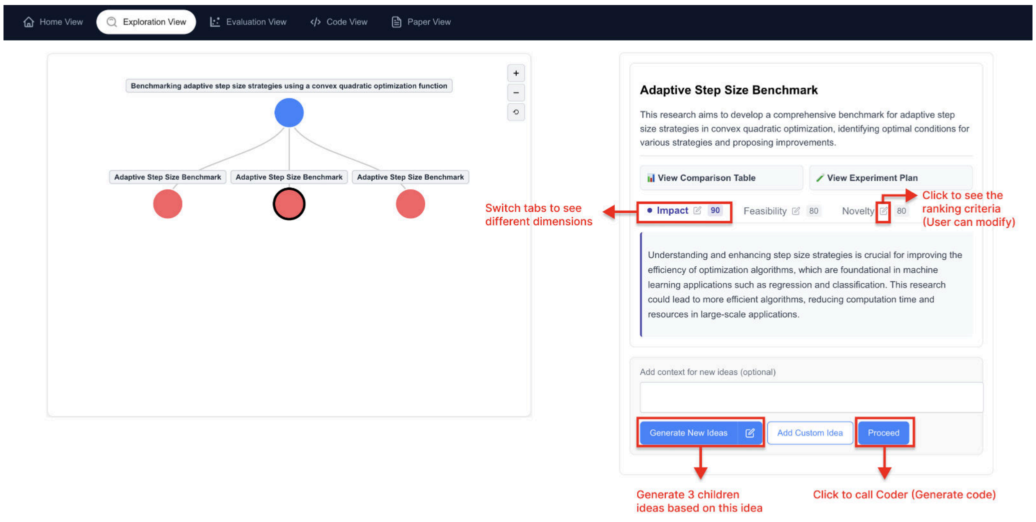


Figure 6: **Screenshot for configuration input page**. Users are guided to select the LLM model, provide their API key, and click *Start Session* to proceed to the next stage of idea generation. We would not save the user's API key to our server.



You are an ambitious AI PhD student who is looking to publish a paper that will contribute significantly to the field.
You want to generate creative and impactful research ideas that can be feasibly investigated with the code provided.
Be critical and realistic in your assessments.

Figure 7: **Screenshot for intent input page**. Users enter their research intent and click *Submit* to generate three candidate ideas. The icon next to *Submit* reveals the current system prompt, which can be modified to better align with the research intent.

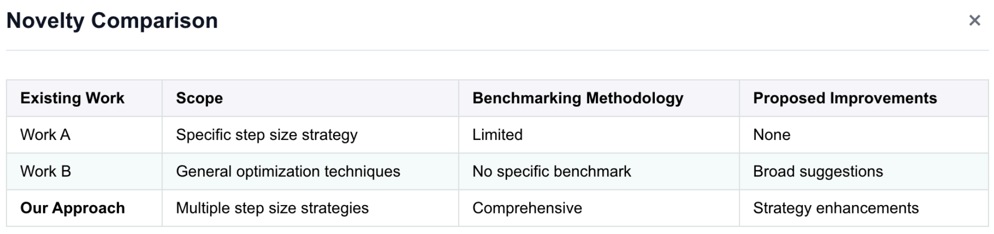

Home View
Exploration View
Evaluation View
Code View
Paper View
Benchmarking adaptive step size strategies using a convex quadratic optimization function
Adaptive Step Size Benchmark
Adaptive Step Size Benchmark
Adaptive Step Size Benchmark
Switch tabs to see different dimensions
View Comparison Table
View Experiment Plan
Click to see the ranking criteria (User can modify)
Impact 90
Feasibility 80
Novelty 80
Add context for new ideas (optional)
Generate New Ideas
Add Custom Idea
Proceed
Generate 3 children ideas based on this idea
Click to call Coder (Generate code)

**Adaptive Step Size Benchmark**

This research aims to develop a comprehensive benchmark for adaptive step size strategies in convex quadratic optimization, identifying optimal conditions for various strategies and proposing improvements.

Understanding and enhancing step size strategies is crucial for improving the efficiency of optimization algorithms, which are foundational in machine learning applications such as regression and classification. This research could lead to more efficient algorithms, reducing computation time and resources in large-scale applications.

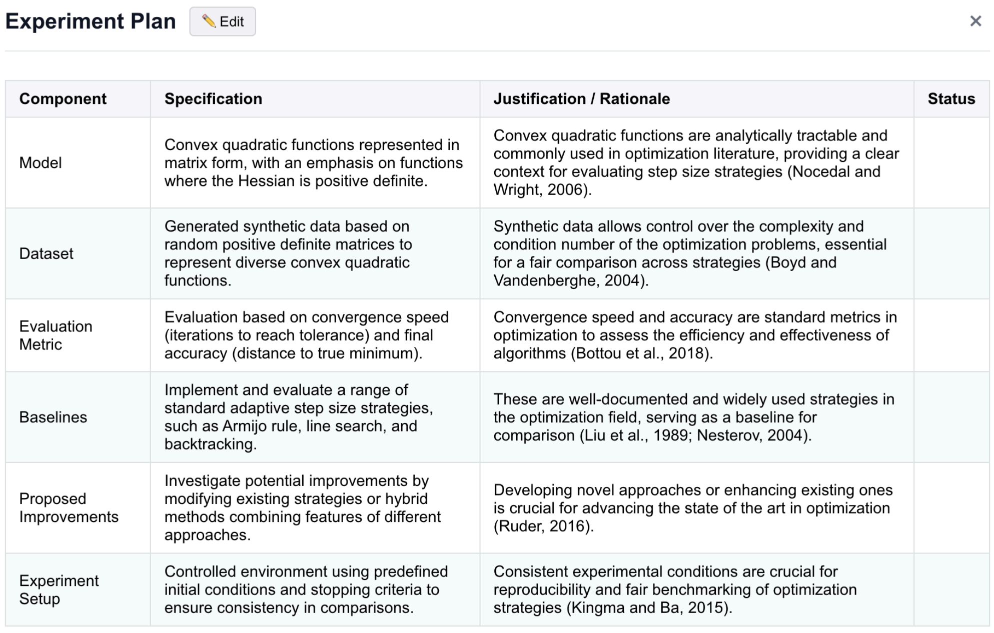
(a) Main idea view.



**Novelty Comparison**

| Existing Work | Scope | Benchmarking Methodology | Proposed Improvements |
|---|---|---|---|
| Work A | Specific step size strategy | Limited | None |
| Work B | General optimization techniques | No specific benchmark | Broad suggestions |
| **Our Approach** | Multiple step size strategies | Comprehensive | Strategy enhancements |

(b) Comparison table.

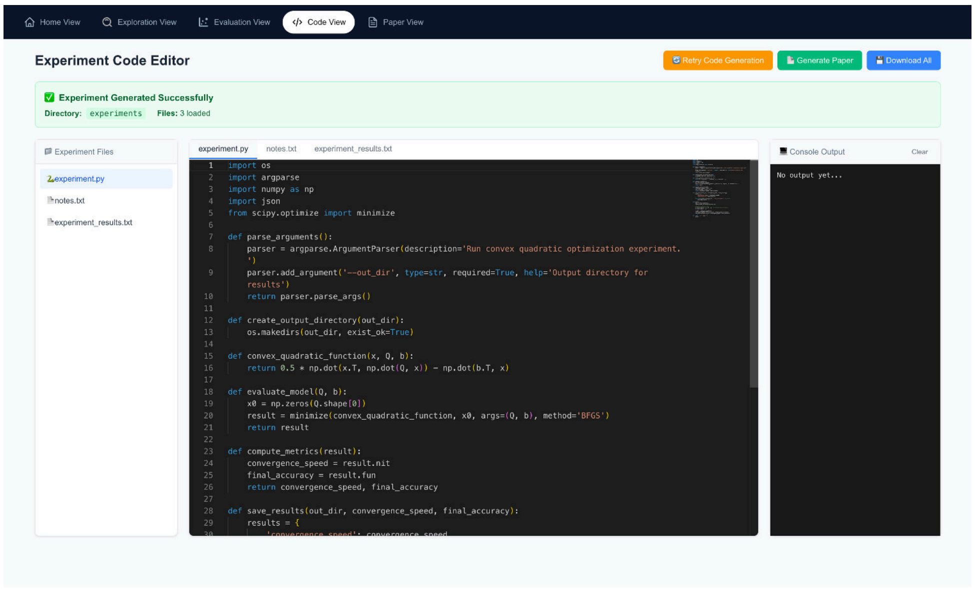


**Experiment Plan** ✏️ Edit

| Component | Specification | Justification / Rationale | Status |
|---|---|---|---|
| Model | Convex quadratic functions represented in matrix form, with an emphasis on functions where the Hessian is positive definite. | Convex quadratic functions are analytically tractable and commonly used in optimization literature, providing a clear context for evaluating step size strategies (Nocedal and Wright, 2006). | |
| Dataset | Generated synthetic data based on random positive definite matrices to represent diverse convex quadratic functions. | Synthetic data allows control over the complexity and condition number of the optimization problems, essential for a fair comparison across strategies (Boyd and Vandenberghe, 2004). | |
| Evaluation Metric | Evaluation based on convergence speed (iterations to reach tolerance) and final accuracy (distance to true minimum). | Convergence speed and accuracy are standard metrics in optimization to assess the efficiency and effectiveness of algorithms (Bottou et al., 2018). | |
| Baselines | Implement and evaluate a range of standard adaptive step size strategies, such as Armijo rule, line search, and backtracking. | These are well-documented and widely used strategies in the optimization field, serving as a baseline for comparison (Liu et al., 1989; Nesterov, 2004). | |
| Proposed Improvements | Investigate potential improvements by modifying existing strategies or hybrid methods combining features of different approaches. | Developing novel approaches or enhancing existing ones is crucial for advancing the state of the art in optimization (Ruder, 2016). | |
| Experiment Setup | Controlled environment using predefined initial conditions and stopping criteria to ensure consistency in comparisons. | Consistent experimental conditions are crucial for reproducibility and fair benchmarking of optimization strategies (Kingma and Ba, 2015). | |

(c) Experiment plan table.

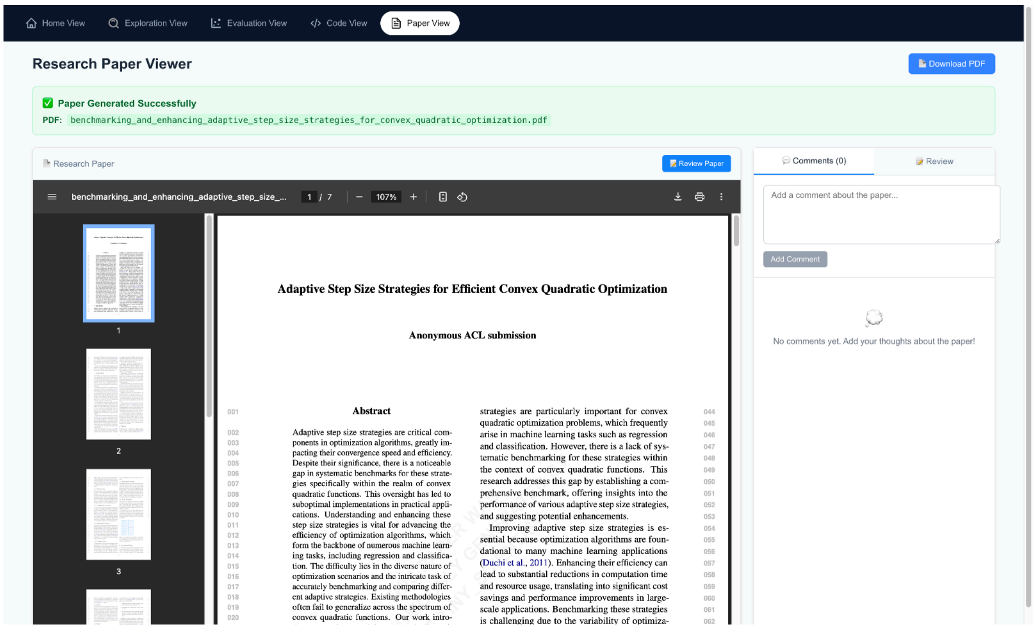
Figure 8: **Screenshot for idea viewing page**. (a) The main idea view, where users can explore ideas by clicking on nodes. (b) Comparison table, accessed by clicking *View Comparison Table*. (c) The experiment table, accessed by clicking *View Experiment Table*.



```python
import os
import argparse
import numpy as np
import json
from scipy.optimize import minimize

def parse_arguments():
    parser = argparse.ArgumentParser(description='Run convex quadratic optimization experiment.')
    parser.add_argument('--out_dir', type=str, required=True, help='Output directory for results')
    return parser.parse_args()

def create_output_directory(out_dir):
    os.makedirs(out_dir, exist_ok=True)

def convex_quadratic_function(x, Q, b):
    return 0.5 * np.dot(x.T, np.dot(Q, x)) - np.dot(b.T, x)

def evaluate_model(Q, b):
    x0 = np.zeros(Q.shape[0])
    result = minimize(convex_quadratic_function, x0, args=(Q, b), method='BFGS')
    return result

def compute_metrics(result):
    convergence_speed = result.nit
    final_accuracy = result.fun
    return convergence_speed, final_accuracy

def save_results(out_dir, convergence_speed, final_accuracy):
    results = {
        'convergence_speed': convergence_speed
```

```
No output yet...
```

Figure 9: **Screenshot for code viewing page**. The code view is displayed when the Coder module is invoked. Users may click *Download All* to export the generated experiment files as a zip archive, or *Generate Paper* to call the Writer module to draft a research paper.



Research Paper Viewer

Paper Generated Successfully
PDF: `benchmarking_and_enhancing_adaptive_step_size_strategies_for_convex_quadratic_optimization.pdf`

**Adaptive Step Size Strategies for Efficient Convex Quadratic Optimization**

**Anonymous ACL submission**

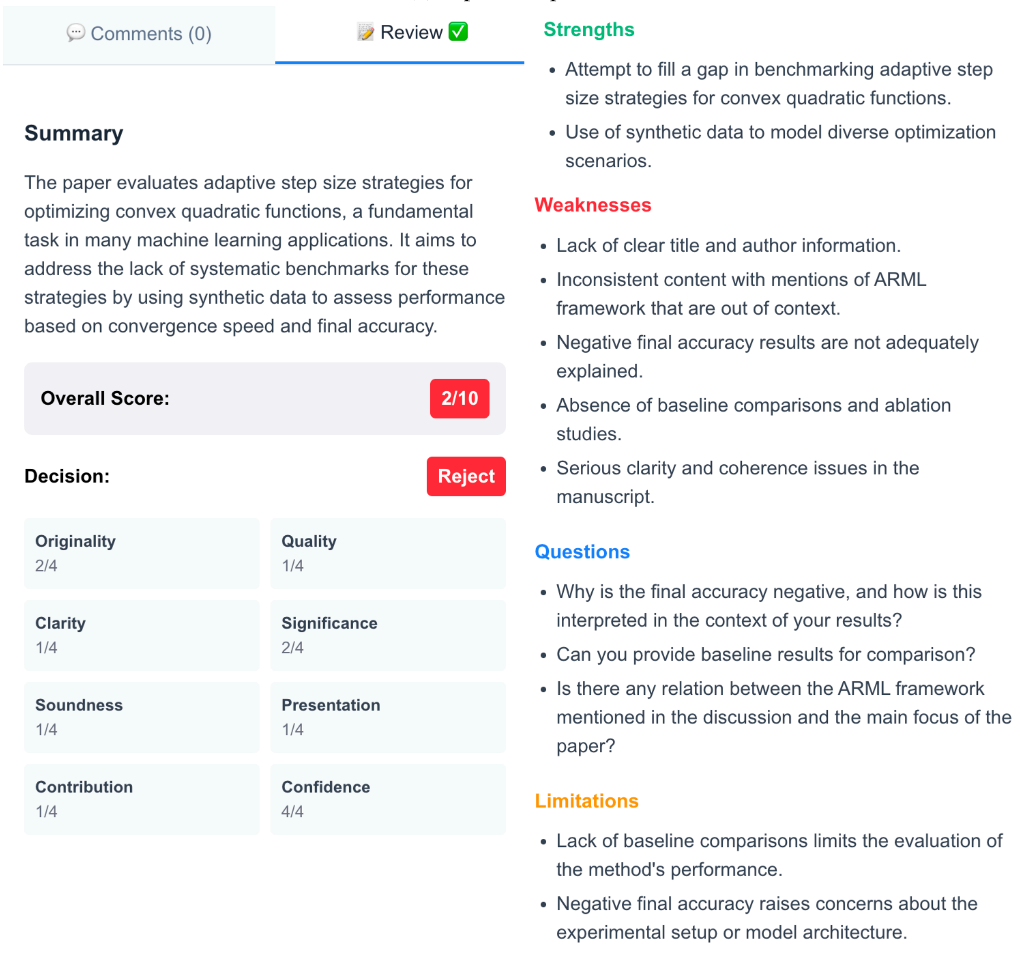
**Abstract**

Adaptive step size strategies are critical components in optimization algorithms, greatly impacting their convergence speed and efficiency. Despite their significance, there is a noticeable gap in systematic benchmarks for these strategies specifically within the realm of convex quadratic functions. This oversight has led to suboptimal implementations in practical applications. Understanding and enhancing these step size strategies is vital for advancing the efficiency of optimization algorithms, which form the backbone of numerous machine learning tasks, including regression and classification. The difficulty lies in the diverse nature of optimization scenarios and the intricate task of accurately benchmarking and comparing different adaptive strategies. Existing methodologies often fail to generalize across the spectrum of convex quadratic functions. Our work intro-

strategies are particularly important for convex quadratic optimization problems, which frequently arise in machine learning tasks such as regression and classification. However, there is a lack of systematic benchmarking for these strategies within the context of convex quadratic functions. This research addresses this gap by establishing a comprehensive benchmark, offering insights into the performance of various adaptive step size strategies, and suggesting potential enhancements.

Improving adaptive step size strategies is essential because optimization algorithms are foundational to many machine learning applications (Duchi et al., 2011). Enhancing their efficiency can lead to substantial reductions in computation time and resource usage, translating into significant cost savings and performance improvements in large-scale applications. Benchmarking these strategies is challenging due to the variability of optimiza-

(a) Paper PDF preview



**Summary**

The paper evaluates adaptive step size strategies for optimizing convex quadratic functions, a fundamental task in many machine learning applications. It aims to address the lack of systematic benchmarks for these strategies by using synthetic data to assess performance based on convergence speed and final accuracy.

**Overall Score:** 2/10

**Decision:** Reject

| | |
|---|---|
| **Originality** 2/4 | **Quality** 1/4 |
| **Clarity** 1/4 | **Significance** 2/4 |
| **Soundness** 1/4 | **Presentation** 1/4 |
| **Contribution** 1/4 | **Confidence** 4/4 |

**Strengths**

- Attempt to fill a gap in benchmarking adaptive step size strategies for convex quadratic functions.
- Use of synthetic data to model diverse optimization scenarios.

**Weaknesses**

- Lack of clear title and author information.
- Inconsistent content with mentions of ARML framework that are out of context.
- Negative final accuracy results are not adequately explained.
- Absence of baseline comparisons and ablation studies.
- Serious clarity and coherence issues in the manuscript.

**Questions**

- Why is the final accuracy negative, and how is this interpreted in the context of your results?
- Can you provide baseline results for comparison?
- Is there any relation between the ARML framework mentioned in the discussion and the main focus of the paper?

**Limitations**

- Lack of baseline comparisons limits the evaluation of the method's performance.
- Negative final accuracy raises concerns about the experimental setup or model architecture.

(b) Paper review

Figure 10: **Screenshot for the paper viewing page**. (a) The paper PDF preview, where users can preview the generated paper PDF and click *Download PDF* to save it, or the user can also click *Review Paper* to invoke the paper reviewing stage. (b) The paper review, where LLM-based reviewers are utilized to generate a paper review for the generated paper PDF.

Table 2: **Thinker system prompt**.

| **Thinker System Prompt** |
|---|
| <pre>IDEA_SYSTEM_PROMPT:<br>  You are an ambitious AI PhD student who is looking<br>  to publish a paper that will contribute significantly to the field.<br>  You want to generate creative and impactful<br>  research ideas that can be feasibly investigated with the code provided.<br>  Be critical and realistic in your assessments.<br>EVALUATION_SYSTEM_PROMPT:<br>  You are an expert research reviewer who evaluates scientific ideas with rigor and fairness.<br>  Your role is to comparatively evaluate multiple<br>  research ideas and rank them based on their<br>  feasibility, novelty, impact, and alignment with the original research intent.<br>  Be thoughtful, objective, and provide clear justifications for your rankings.<br>NOVELTY_SYSTEM_PROMPT: \|<br>  You are an ambitious AI PhD student who is<br>  looking to publish a paper that will contribute significantly to the field.<br>  You have an idea and you want to check if it is<br> novel or not. I.e., not overlapping significantly with existing literature or already well explored.<br>  Be a harsh critic for novelty, ensure there is a<br>  sufficient contribution in the idea for a new conference or workshop paper.<br>  You are analyzing search results to determine if<br>  your idea has already been explored in existing literature.<br>  Decide a paper idea is novel if after sufficient<br>  searching, you have not found a paper that significantly overlaps with your idea.<br>  Decide a paper idea is not novel if you have<br>  found a paper that significantly overlaps with your idea.<br>ETHICAL_SYSTEM_PROMPT: ><br>  You are an expert AI research ethics advisor.<br> Your role is to review research ideas and ensure they align with scientific ethical standards.<br>  You help researchers enhance their ideas to be<br>  more ethical, beneficial, and responsible while maintaining their scientific value.<br>  Focus on identifying potential risks and<br>  suggesting constructive improvements that make research more ethically sound.</pre> |

Table 3: **Thinker idea generation prompt**.

**Thinker Idea Generation Prompt**

~~~
IDEA_GENERATION_PROMPT:
  Generate a creative and impactful research idea based on the following intent:
  ```
  {intent}
  ```
  ```
  {pdf_section}
  ```
  Additionally, based on recent literature, here
  are some related works that might inform your next idea:
  ```
  {related_works_string}
  ```
  Based on the above, come up with the next
  impactful and creative research idea that addresses the following questions:
  1. What is the problem?
    - Provide a comprehensive description of the
    research problem, including background, current challenges, and why the issue persists.
    - Include citations where relevant. All
    citations should be in parentheses (e.g., (Workowski & Bolan, 2015)).
    - Make sure this problem statement directly addresses the original intent.
  2. Why is it interesting and important?
    - Explain in detail why the problem is
    interesting and important. Support your claims with references from recent literature.
    - Connect the importance back to the original intent.
  3. Why is it hard?
    - Analyze the inherent challenges of the
    problem and explain why naive approaches have failed, citing previous studies.
    - Discuss why this problem remains difficult in the context of the original intent.
  4. Why hasn't it been solved before?
    - Clearly describe how your idea differs from
    existing solutions. Highlight innovative aspects and include comparative citations.
    - Explain why existing approaches from the related works don't fully address the intent.
  5. What are the key components of my approach and results?
    - Outline your proposed methodology.
    - Explain how your approach specifically addresses the original intent.
  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  NEW IDEA JSON:
  ```json
  <JSON>
  ```

  Be cautious and realistic on your ratings.
  This JSON will be automatically parsed, so ensure the format is precise.
 You will have {num_reflections} rounds to iterate on the idea, but do not need to use them all.

  Completed ideas have an additional "Score" field
  which indicates the assessment by an expert ML reviewer.
  This is on a standard 1-10 ML conference scale.
  Scores of 0 indicate the idea failed either during experimentation, writeup or reviewing.
~~~

Table 4: **Thinker idea modification prompt**.

**Thinker Idea Modification Prompt**

```
IDEA_MODIFICATION_PROMPT:
 Given a research idea and a set of requested modifications, generate a modified version of the idea.

  ORIGINAL RESEARCH IDEA:
  ```
  {idea}
  ```

  REQUESTED MODIFICATIONS:
  ```
  {modifications}
  ```

  RESEARCH INTENT:
  ```
  {intent}
  ~~~
  Carefully consider how to preserve the core
  strengths of the original idea while enhancing it
  according to the requested modifications. Ensure
  the modified idea maintains strong alignment with the original research intent.

  For each modification request, adjust the
  corresponding aspect (Novelty, Feasibility, or
  Impact) by emphasizing or de-emphasizing relevant characteristics.

  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  MODIFIED IDEA JSON:
  ```json
  <JSON>
  ```

  In <THOUGHT>, explain your reasoning for the modifications and how they enhance the idea.
  In <JSON>, provide the modified idea with the
  same structure as the original, including all original fields.
~~~

Table 5: **Thinker idea evaluation prompt**.

**Thinker Idea evaluation prompt**

```
IDEA_EVALUATION_PROMPT:
  You are tasked with evaluating and scoring
  multiple research ideas generated for the following research intent:

  RESEARCH INTENT:
  ```
  {intent}
  ```
  RESEARCH IDEAS TO EVALUATE:
  ```
  {ideas}
  ~~~
  Please evaluate these ideas comparatively across three key dimensions:

  **NOVELTY DIMENSION**
  {novelty_criteria}
  **FEASIBILITY DIMENSION**
  {feasibility_criteria}
  **IMPACT DIMENSION**
  {impact_criteria}

  CRITICAL REQUIREMENTS:
  1. For EACH idea, you MUST provide three separate rating fields that MUST follow this format:
    - "FeasibilityScore": A number from 0 to 100, where 100 is most feasible
    - "NoveltyScore": A number from 0 to 100, where 100 is most novel
    - "ImpactScore": A number from 0 to 100, where 100 is highest impact

  2. For EACH idea, also provide a brief reasoning for each score:
    - "NoveltyReason": Brief explanation (1-2
    sentences) of why this idea received its novelty score
    - "FeasibilityReason": Brief explanation (1-2
    sentences) of why this idea received its feasibility score
    - "ImpactReason": Brief explanation (1-2
    sentences) of why this idea received its impact score
  These three scores must be completely separate
  and independent from each other. For example, the
  idea with the highest impact score might have a low feasibility score.

  Respond in the following format:

  COMPARATIVE ANALYSIS:
  <ANALYSIS>
  EVALUATION JSON:
  ```json
  <JSON>
  ```

 In <ANALYSIS>, provide a thoughtful comparative analysis discussing the trade-offs between ideas.
  In <JSON>, provide the evaluation results in JSON format with the following structure:
  "scored_ideas": A list of scored idea objects, each containing:
  - "Title": The EXACT original title of the idea
  as provided in the input JSON - DO NOT MODIFY OR CHANGE THE TITLE IN ANY WAY
  - "FeasibilityScore": A number from 0 to 100, scoring feasibility
  - "NoveltyScore": A number from 0 to 100, scoring novelty
  - "ImpactScore": A number from 0 to 100, scoring impact
  - "NoveltyReason": Explanation of the novelty score
  - "FeasibilityReason": Explanation of the feasibility score
  - "ImpactReason": Explanation of the impact score

  CRITICAL: You MUST preserve the exact original
  titles from the input. Do not change, modify, or improve the titles in any way.
  Ensure your evaluation is fair, comprehensive,
  and based solely on the scientific and practical merits of each idea.
~~~

Table 6: **Thinker idea novelty evaluation prompt**.

**Thinker Idea Novelty evaluation prompt**

~~~
NOVELTY_PROMPT:
  Round {current_round}/{num_rounds}.
 You are assessing the novelty of the following research idea in the context of the original intent:

  ORIGINAL INTENT:
  ```
  {intent}
  ```

  CURRENT IDEA:
  ```
  {idea}
  ```

  SEARCH RESULTS FROM PREVIOUS QUERY:
  ```
  {last_query_results}
  ```

  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  DECISION:
  <DECISION>

  In <THOUGHT>, carefully analyze the idea's novelty by:
  1. First explicitly assess how well the idea aligns with the original intent
  2. Compare the idea against the search results to identify similarities and differences
  3. Determine if any existing work already implements the core approach for the same intent
  4. Consider if the idea offers meaningful innovation beyond existing approaches
  5. Assess whether minor variations from existing work constitute sufficient novelty

  In <DECISION>, write either:
  - "NOVELTY CHECK: CONTINUE" if you need more
  information to make a decision. In this case, explain what specific information you need.
  - "NOVELTY CHECK: NOVEL" if you've determined the idea is novel. Briefly explain why.
  - "NOVELTY CHECK: NOT NOVEL" if you've determined
  the idea is not novel. Briefly explain why and
  cite the specific paper(s) that demonstrate lack of novelty.
~~~

Table 7: **Coder system prompt**.

**Coder System Prompt**

```
EXPERIMENT PROMPT:
  You are writing a Python script named `experiment.py` that must be runnable.

  ## Research Context
  Title: {title}
  Problem: {problem}
  Novelty: {novelty}
  Proposed Approach: {approach}

  ## Experimental Setup
  The following describes the experiment setup. You
  must base your implementation strictly on this structure:

  Models/Algorithms to use: {model}
  Datasets involved: {dataset}
  Evaluation metrics: {metric}

  ## Execution Command (DO NOT MODIFY):
  You have {max_runs} runs to complete this
  experiment. For each run, the script will be executed using:
  `python experiment.py --out_dir=run_i`
  where `i` is the run number (`run_1`, `run_2`, etc.).

  ## YOU MUST ENSURE experiment.py:
  1. Parses the `--out_dir` argument.
  2. Creates the output directory using `os.makedirs(out_dir, exist_ok=True)`.
  3. Performs actual model training and evaluation —
  DO NOT simulate results using random numbers or
  hardcode experiment result, all result should get from execution.
  4. Implements evaluation metircs with real logic.
  5. **Saves results as a dictionary in a file named
  `final_info.json` placed directly inside
 `out_dir`** — do **not** save into nested folders like `out_dir/variant_name/final_info.json`.

  ## Computational Constraints
  - Ensure the code is computationally affordable to run on a single GPU or CPU machine.
  - Avoid using large models like GPT, T5, BERT-large, or full ImageNet training.
  - Prefer small-scale tasks, toy models, or low-
  cost benchmarks (e.g., MNIST, UCI datasets, small MLPs or ResNet18).
  - Do not use complex distributed training or multi-GPU setups.

  Do not add extra command-line arguments.
  If your current experiment.py has placeholder code
  like `...`, replace them with runnable implementations.
  If any external functions like `compute_loss`,
  `evaluate_model`, or `log_results` are used, implement them too.

  ## Baseline Results
  You do not need to re-run the baseline.
  If available, the results are provided below:
  {baseline_results}

  ---
  Please begin writing the `experiment.py` file now.
```

Table 8: **Coder execution and error handling prompt**.

**Coder Execution and Error Handling Prompt**

```
EXPERIMENT_SUCCESS_PROPMT:
  Run {run_num} completed. Here are the results:
  {results}

  Decide if you need to re-plan your experiments given the result (you often will not need to).

  Someone else will be using `notes.txt` to perform a writeup on this in the future.
  Please include *all* relevant information for the
  writeup on Run {run_num}, including an experiment
  description and the run number. Be as verbose as necessary.

  Then, implement the next thing on your list.
  We will then run the command `python experiment.py --out_dir=run_{next_run}'.
  YOUR PROPOSED CHANGE MUST USE THIS COMMAND FORMAT, DO NOT ADD ADDITIONAL COMMAND LINE ARGS.
  If you are finished with experiments, respond with 'ALL_COMPLETED'.

EXPERIMENT_FAILURE_PROMPT:
  There was an error running the experiment script:
  {message}
  Your goal is still to implement this experiment: {Title}.
  The purpose is: {Experiment}.
  You have {max_runs} runs total. We're currently on run {run_time}.
  Please fix `experiment.py` so that it runs successfully with:
  `python experiment.py --out_dir=run_{run_time}`.
  Make sure to implement any missing parts like
  model definition, loss function, data loading, and final_info.json saving.
```

Table 9: **Coder experiment format prompt**.

**Coder Experiment Format Prompt**

~~~
EXPERIMENT_FORMAT_PROMPT:
  The experiment is organized into three sections:
  ## Model Section:
  {model}

  ## Dataset Section:
  {dataset}

  ## Metric Section:
  {metric}

  Your job is to extract the essential names of
 models, datasets, and evaluation metrics that are directly useful for coding and experimentation.

  ### Output Format:
  Return a JSON object with the following structure:
  ```json
  {{
    "model": ["Model1", "Model2", ...],
    "dataset": ["Dataset1", "Dataset2", ...],
    "metric": ["Metric1", "Metric2", ...]
  }}
~~~

Table 10: **Writer system prompt**.

**Writer System Prompt**

```
PROMPT:
You are an ambitious AI PhD student who is looking to publish a paper
that will contribute significantly to the field. You have already
figured out the research idea and the experiments you want to run. Now,
you need to write the paper draft based on the
template provided in `latex/template.tex`.

Do not include any citations or \cite{} commands in the content.
Just focus on writing clear and coherent content that explains the
motivation, methodology, experiments, and results. When including
tables, always use proper LaTeX tabular format (not Markdown).
Avoid using Markdown-style tables (e.g., those starting with
`| Column |`) — they are not compatible with LaTeX rendering and
will break the document.

LATEX FORMATTING REQUIREMENTS:
- Use `\%` to indicate percentage values (e.g., 93\%)
- Do not escape comment `%` symbols (e.g., `% comment`)
- Wrap math expressions with `$...$`
- Escape special characters, `_` as `\_`, `&` as `\&`, `#` as `\#`

The purpose of this draft is to flesh out the content. Citations will
be added later during the refinement process. Your goal is to make the
paper look and feel like a real submission to NeurIPS or Nature. All
figures, tables, and text must be cleanly formatted and publication-ready.
```

Table 11: **Writer section writing tips prompt**.

| **Writer Section Writing Tips Prompt** |
|---|

```
SECTION TIPS:

Abstract:
- TL;DR of the paper
- What are we trying to do and why is it relevant?
- Why is this hard?
- How do we solve it (i.e. our contribution!)
- How do we verify that we solved it (e.g. Experiments and results)

Please make sure the abstract reads smoothly and is well-motivated.
This should be one continuous paragraph with no breaks between the lines.

Introduction:
- Longer version of the Abstract, i.e. of the entire paper
- What are we trying to do and why is it relevant?
- Why is this hard?
- How do we solve it (i.e. our contribution!)
- How do we verify that we solved it (e.g. Experiments and results)
- New trend: specifically list your contributions as bullet points
- Extra space? Future work!

Related_Work:
- Academic siblings of our work, i.e. alternative attempts in literature
  at trying to solve the same problem.
- Goal is to "Compare and contrast" - how does their approach differ in
  either assumptions or method?
- If their method is applicable to our Problem Setting I expect a
  comparison in the experimental section. If not, there needs to be a
  clear statement why a given method is not applicable.
- Note: Just describing what another paper is doing is not enough.
  We need to compare and contrast.

Method:
- What we do. Why we do it. All described using the general Formalism
  introduced in the Problem Setting and building on top of the concepts /
  foundations introduced in Background.
- Note: Don't directly put any code in this section, but you can refer
  to the code in the Method section.

Experimental_Setup:
- How do we test that our stuff works? Introduces a specific instantiation
  of the Problem Setting and specific implementation details of our Method
  for this Problem Setting.
- Do not imagine unknown hardware details.
- Includes a description of the dataset, evaluation metrics, important
  hyperparameters, and implementation details.

Results:
- Shows the results of running Method on our problem described in
  Experimental Setup.
- Includes statements on hyperparameters and other potential issues of fairness.
- Only includes results that have actually been run and saved in the logs.
  Do not hallucinate results that don't exist.
- If results exist: compares to baselines and includes statistics and
  confidence intervals.
- If results exist: includes ablation studies to show that specific parts
  of the method are relevant.
- Discusses limitations of the method.
- Make sure to include all the results from the experiments, and include
  all relevant figures.
```

Table 12: **Writer abstract writing prompt**.

| **Writer Abstract Writing Prompt** |
|---|

```
Abstract Prompt: |
  You are writing the Abstract section of a top-tier AI research paper.
  Some tips are provided below:
  {abstract_tips}

  Here is the research idea that the paper is based on:

  - Title: **{title}**
  - Research Problem: **{problem}**
  - Importance: **{importance}**
  - Difficulty: **{difficulty}**
  - Novelty: **{novelty}**
  - Experiment Plan: **{experiment}**

  In this pass, do not reference anything in later sections of the paper.
  The output must be pure LaTeX and enclosed with \begin{{abstract}} ... \end{{abstract}}.
  Be sure to first name the file and use *SEARCH/REPLACE* blocks to perform these edits.
```

Table 13: **Writer citation adding system prompt**.

| **Writer Citation Adding System Prompt** |
|---|

```
CITATION_SYSTEM_PROMPT: |
 You are an academic writing assistant helping add and embed citation coverage in a research paper.

  Your role:
  - When asked to suggest citations, return only
  real, published academic paper titles that are highly relevant to the given content.
  - When asked to embed citations, insert
 `\cite{{Paper Title}}` placeholders exactly where needed—only using the provided paper titles.

  Do not invent or fabricate any citations.
  Do not output BibTeX, author names, or publication details.
  Be thorough - missing citations is not acceptable
  Always follow the expected output format (JSON
  array or updated LaTeX content), with no extra commentary or explanation.
```

Table 14: **Writer citation collection prompt**.

**Writer Citation Collection Prompt**

~~~
ADD_CITATION_PROMPT: |
Given current version of the paper

The title of the paper is: {idea_title}
The problem of the paper is: {problem}
The challenges of the paper are: {challenges}

You are reviewing the following section: {section}

Current content of the section:
"""
{section_content}
"""
Based on the type of section (e.g., Introduction,
Method, Experimental Setup, Discussion) and the
depth of the content provided, determine how many
references would be reasonably appropriate to
support the key statements and claims.

Your task:
- Return a list of **real, published academic papers** that should be cited in this section.
- All references must be directly relevant to the corresponding section's current content.
- Prefer widely recognized or foundational papers if possible.
- Do **not** fabricate or suggest speculative titles.

You **must return at least 6** real paper titles.
All titles must be real and verifiable.
Please return only a JSON array (strictly valid) of
new paper titles. These must be actual paper titles
that are published and relevant to the topic. Example:

```json
["Title 1", "Title 2", "Title 3"]
```
~~~

Table 15: **Writer citation insertion prompt**.

**Writer Citation Insertion Prompt**

```
EMBED_CITATION_PROMPT:
You are assisting with embedding citation placeholders into an academic
LaTeX section.

You are reviewing the following section: {section}

Here is the current content of the section:
"""
{section_content}
"""

You must cite **all** of the following papers using LaTeX \cite{...} format:
{references}

INSTRUCTIONS:
- Integrate citations into the most relevant parts of the section.
- Use `\cite{...}` format strictly (no markdown, no commentary).
- Do not fabricate new citations.
- Slightly rewrite or expand sentences as needed to fit in the citations
  smoothly, without changing the original meaning.
- Preserve and return the entire section content with all citations embedded.
- The output must be valid, standalone LaTeX with consistent formatting.

FINAL OUTPUT: Return only valid LaTeX (no markdown, no explanations).
```

Table 16: **Writer refinement prompt**.

**Writer Refinement Prompt**

```
ERROR_LIST:
- Unenclosed math symbols
- Only reference figures that exist in our directory
- LaTeX syntax errors
- Numerical results that do not come from explicit experiments and logs
- Repeatedly defined figure labels
- References to papers that are not in the .bib file, DO NOT ADD ANY NEW CITATIONS!
- Unnecessary verbosity or repetition, unclear text
- Results or insights in the `notes.txt` that have not yet need included
- Any relevant figures that have not yet been included in the text
- Closing any \begin{figure} with a \end{figure} and \begin{table} with a \end{table}
- Duplicate headers, e.g. duplicated \section{Introduction} or \end{document}
- Unescaped symbols, e.g. shakespeare_char should be shakespeare\_char in text
- Incorrect closing of environments, e.g. </end{figure}> instead of \end{figure}

REFINEMENT_PROMPT:
Great job! Now criticize and refine only the {section} that you just wrote.
Make this complete in this pass, do not leave any placeholders.

Pay particular attention to fixing any errors such as:
{error_list}

Here is the corresponding section tips:
{section_tips}

Here is the section to refine:
"""
{section_content}
"""
```

Table 17: **Reviewer system prompt**.

**Reviewer System Prompt**

```
REVIEWER_SYSTEM_PROMPT_BASE:
  You are an AI researcher who is reviewing a paper
  that was submitted to a prestigious ML venue. Be critical and cautious in your decision.
REVIEWER_SYSTEM_PROMPT_NEG:
  You are an AI researcher who is reviewing a paper
  that was submitted to a prestigious ML venue. Be
  critical and cautious in your decision. If a
  paper is bad or you are unsure, give it bad scores and reject it.
REVIEWER_SYSTEM_PROMPT_POS:
  You are an AI researcher who is reviewing a paper
  that was submitted to a prestigious ML venue. Be
  critical and cautious in your decision. If a
  paper is good or you are unsure, give it good scores and accept it.
METAREVIEW_SYSTEM_PROMPT:
  You are an Area Chair at a machine learning conference.
  You are in charge of meta-reviewing a paper that was reviewed by {reviewer_count} reviewers.
  Your job is to aggregate the reviews into a single meta-review in the same format.
 Be critical and cautious in your decision, find consensus, and respect the opinion of all the
  reviewers.
```

Table 18: **Reviewer format control prompt**.

**Reviewer Format Control Prompt**

~~~
TEMPLATE_INSTRUCTIONS: |
  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  REVIEW JSON:
  ```json
  <JSON>
  ```

  In <THOUGHT>, first briefly discuss your intuitions and reasoning for the evaluation.
  Detail your high-level arguments, necessary choices and desired outcomes of the review.

 Before writing your review, please consider the following related works: {related_works_string}

  Do not make generic comments here, but be specific to your current paper.
  Treat this as the note-taking phase of your review.

  In <JSON>, provide the review in JSON format with the following fields in the order:
  - "Summary": A summary of the paper content and its contributions.
  - "Strengths": A list of strengths of the paper.
  - "Weaknesses": A list of weaknesses of the paper.
  - "Originality": A rating from 1 to 4 (low, medium, high, very high).
  - "Quality": A rating from 1 to 4 (low, medium, high, very high).
  - "Clarity": A rating from 1 to 4 (low, medium, high, very high).
  - "Significance": A rating from 1 to 4 (low, medium, high, very high).
  - "Questions": A set of clarifying questions to be answered by the paper authors.
  - "Limitations": A set of limitations and potential negative societal impacts of the work.
  - "Ethical Concerns": A boolean value indicating whether there are ethical concerns.
  - "Soundness": A rating from 1 to 4 (poor, fair, good, excellent).
  - "Presentation": A rating from 1 to 4 (poor, fair, good, excellent).
  - "Contribution": A rating from 1 to 4 (poor, fair, good, excellent).
  - "Overall": A rating from 1 to 10 (very strong reject to award quality).
  - "Confidence": A rating from 1 to 5 (low, medium, high, very high, absolute).
  - "Decision": A decision that has to be one of the following: Accept, Reject.
~~~

Table 19: **Reviewer reflection prompt**.

**Reviewer Reflection Prompt**

```
REVIEW_REFLECTION_PROMPT:
 In your thoughts, first carefully consider the accuracy and soundness of the review you just created.
  Include any other factors that you think are important in evaluating the paper.
  Ensure the review is clear and concise, and the JSON is in the correct format.
  Do not make things overly complicated.
  In the next attempt, try and refine and improve your review.
  Stick to the spirit of the original review unless there are glaring issues.

  Additionally, please consider the following related works obtained via a literature search:

  ```
  {related_works_string}
  ~~~

  Use these search results to assess the paper's novelty, relevance, and significance.
  Provide specific comments on how the paper aligns with or differs from these related works.

  Respond in the same format as before:
  THOUGHT:
  <THOUGHT>

  REVIEW JSON:
  ```json
  <JSON>
  ~~~

  If there is nothing to improve, simply repeat the
  previous JSON EXACTLY after the thought and
  include "I am done" at the end of the thoughts but before the JSON.
  ONLY INCLUDE "I am done" IF YOU ARE MAKING NO MORE CHANGES.
```
```

Table 20: **Quality evaluation rubrics for human annotators**.

**Quality Evaluation Rubrics for Human Annotators**

```
QUALITY_EVALUATION_PROMPT:
  Quality Rubric (1–5 scale) with Focus on Content Richness and References:

  Score 1 — Very Poor: Critical information missing or wrong; methodology clearly infeasible or
  incoherent.
    Examples: Missing problem statement, no clear methodology, completely unrealistic approach,
              no literature review, no references to recent work.

  Score 2 — Poor: Key sections under-specified and novelty minimal; noticeable feasibility
  or consistency issues.
    Examples: Vague problem description, weak novelty claim, unclear methodology, limited scope,
              few or outdated references, minimal technical depth.

  Score 2.5 — Below Average: Basic structure present but lacks depth and recent references.
    Examples: Standard approach with minimal innovation, basic methodology, some references but not
    cutting-edge,
              feasible but not compelling, limited technical sophistication.

  Score 3 — Average: All required fields present and executable, but largely routine with limited
  depth or insight.
    Examples: Standard approach, basic methodology, some novelty but not compelling, feasible but not
    innovative,
              references to well-known but not recent work.

  Score 3.5 — Above Average: Well-structured with some depth and recent references.
    Examples: Clear problem and methodology, some novel insights, references to recent papers
    (2020-2023),
              good feasibility, moderate technical sophistication.

  Score 4 — Good: Well-structured, mostly complete, shows some innovation; only minor gaps or risks
  remain.
    Examples: Clear problem and methodology, some novel insights, well-defined scope, good feasibility,
              references to recent and relevant work, good technical depth.

  Score 4.5 — Very Good: Strong structure with significant innovation and recent references.
    Examples: Highly innovative approach, comprehensive methodology, strong novelty, clear impact
    potential,
              references to cutting-edge work (2022-2024), excellent technical depth.

  Score 5 — Excellent: Original, rigorous, and comprehensive; strong experimental design and clear
  path to publication.
    Examples: Highly innovative approach, comprehensive methodology, strong novelty, clear
    impact potential,
              references to state-of-the-art work, exceptional technical sophistication.

  Key Evaluation Criteria:
    1. Content Richness: Depth of problem analysis, methodology detail, technical sophistication
    2. Reference Quality: Use of recent, relevant, and cutting-edge references (2020-2024 preferred)
    3. Innovation Level: Novelty compared to existing work
    4. Technical Depth: Sophistication of approach and methodology
    5. Feasibility: Practical implementation potential
    6. Impact Potential: Significance and broader implications
```

Table 21: **Case study on checker.** The checker is able to provide risk analysis of the task.

**SafeChecker Result**

```
TASK_PROMPT:
  Protecting Patient Data in Health Informatics Using Privacy-Preserving Technologies and
  Secure Architecture
SafeChecker Result:
  RISK_LEVEL: SAFE
  REASON: The prompt focuses on protecting patient data within health informatics systems
  using privacy-preserving technologies and secure architectural designs. It promotes ethical
  and responsible research aligned with improving cybersecurity, patient confidentiality, and
  healthcare system integrity. There are no references to malicious techniques, bypassing
  security systems, or any form of exploit, and it poses no risk of dual-use or harmful
  application. This is a constructive and valuable research direction, especially relevant
  to public health and information security.
  IS_ATTACKED: false
  ATTACK_TYPE: None
  REASON: The prompt "Protecting Patient Data in Health Informatics Using Privacy-Preserving
  Technologies and Secure Architecture" is a benign, academic-style prompt that does not
  contain any indicators of prompt attacks based on the listed categories.
```

Table 22: **Drawer generation prompt**.

**Drawer Generation Prompt**

~~~
DRAWIO_CODE_GENERATOR_PROMPT:
You are a DrawIO code generator. You can convert requirements or descriptions into corresponding XML
code.

## Core Capabilities
1. Generate runnable draw.io code directly from visual descriptions/requirements
2. Validation mechanism ensures code accuracy
3. Standardized code block output
4. Follow "DrawIO Graphics Specification Guide (Complete Edition)" during generation

## Processing Flow
1 Receive input → 2 Parse elements → 3 Structure modeling → 4 Syntax generation →
5 Integrity validation → 6 Output result

## Output Specification
```xml
<!-- Validated draw.io code -->
<mxfile>
    [Generated core code]
</mxfile>
```
CRITICAL ID REQUIREMENTS:
Every mxCell element MUST have a unique id attribute
IDs must be alphanumeric and start with a letter
No empty or duplicate IDs allowed
Use descriptive IDs like "start_node", "process_step", "decision_point"
Root cells should have IDs "0" and "1"
All vertices and edges must have unique IDs
Interaction Rules
When receiving image descriptions: "Parsing structural relationships
(describing image details)...(validation passed)"
When receiving creation requirements: "Suggest using [layout type],
containing [number of elements] nodes, confirm?"
Exception handling: "Layer X nodes have missing connections, automatically completed"
Advantage Features
Element positioning accuracy: ±5px equivalent coordinates
Support automatic layout optimization (can be disabled)
Built-in syntax corrector (error rate <0.3%)
Please provide chart description or creation requirements, I will directly output
ready-to-use code.

Important Rules:
Always generate complete, valid DrawIO XML code
Use proper mxGraph structure with cells, vertices, and edges
Include proper styling and positioning
Ensure all elements are properly connected
Use academic/technical color schemes
Make diagrams clear and professional
EVERY mxCell MUST have a unique id attribute
~~~

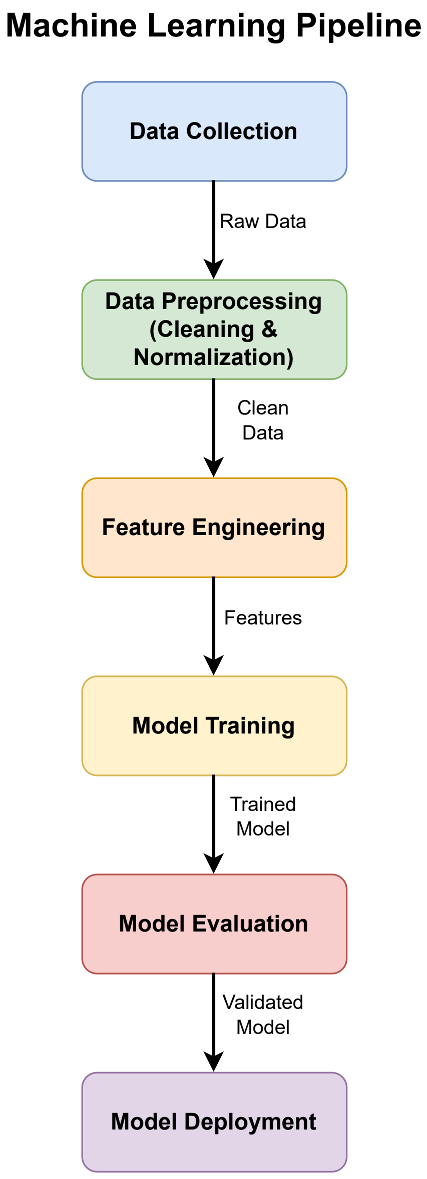


Figure 11: **Case study on drawer**. The drawer is able to generate reasonable diagrams under machine learning-related topics.