LEARNING SCIENTIFIC EXPLORATION FROM HUMAN RESEARCH DECISIONS TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

A key challenge in building AI systems for scientific research is enabling *scientific exploration*: the systematic process of investigating unknown phenomena or ideas to gain new knowledge through sequences of research decisions and actions. Yet this process is largely missing from existing scientific corpora; for example, research papers primarily record final outcomes rather than the trajectories that produced them. In this work, we introduce **ResearchTrails**, a dataset of human research trajectories constructed from Git repositories, where commit histories serve as proxies for research exploration. We develop an automated and scalable pipeline that extracts structured research trajectories from repository commits, capturing successive changes to methods, experiments, and ablations. We characterize the resulting dataset and show that these trajectories contain meaningful signals about intermediate research decisions beyond what final papers reveal. We further demonstrate utilities of ResearchTrails in multiple use cases, including retrieving human research experience as external skills at test time and training models on research trajectories to improve generalization to new research decisions. Our results suggest a path toward AI systems that learn not only from the products of science, but from the evolving process of discovery itself.

1 INTRODUCTION

Existing systems for automating scientific research powered with large language models (LLMs) have demonstrated promising capabilities across many components of the research process, including generating research ideas, synthesizing literature (Asai et al., 2024; Skarlinski et al., 2024), writing code (Yang et al., 2024), and executing other relevant well-specified tasks (Kon et al., 2026; Chan et al., 2025b). Agents have been chained together into agentic workflows to simulate the entire research lifecycle or simulate certain research communities (Yu et al., 2024).

Despite rapid progress in automated research, how models can learn the evolving process of *scientific exploration*, i.e., how to make a sequence of research decisions as a project unfolds, is underexplored. In addition to generating ideas or executing a predefined plan, humans typically perform a series of explorations during the process, such as modifying methods after observing unexpected behavior, introducing new experiments to test hypotheses, and abandoning or revising directions that appear unpromising (Klahr & Dunbar, 1988; Dunbar, 1994).

A central bottleneck of improving LLM's capabilities in scientific exploration is the lack of supervision. Research papers remain the most popular way of sharing findings, but they primarily describe the final outcome of projects. The winding processes that produce the outcomes are absent (Rosenthal, 1979). Recent works have explored agentic research trajectories by letting LLMs interact with synthetic research environments (e.g., other LLMs) (Cai & Behl, 2026; Si et al., 2026b), or synthesizing reasoning traces (Wang et al., 2026). However, these trajectories may not reflect the decision processes of human researchers, which is challenging to obtain at scale. Therefore, we ask: *Can we create human research-decision trajectories at scale? Does learning from those decision trajectories improve scientific exploration beyond learning only from final research outcomes?*

In this work, we introduce ResearchTrails, a dataset of human research trajectories constructed from the development histories of scientific projects. Our key insight is to use Git repositories as proxies of how human research evolves over time. We develop an automated pipeline to transform commit histories into a series of research decisions in the exploration process. Daily commit histories

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

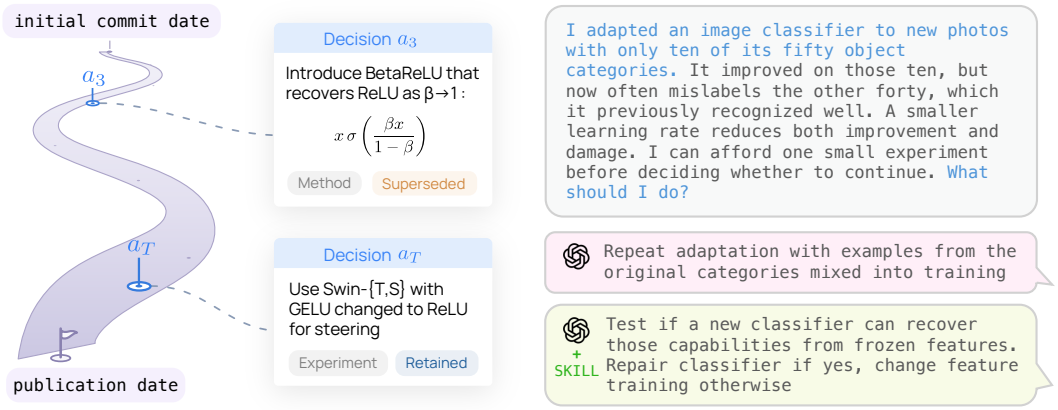


Figure 1: Overview of ResearchTrails: learning scientific exploration from human research trajectories. Left: We extract research decisions from GitHub commit history. Right: We utilize these trajectories to improve LLMs' capability of assisting with research exploration. For example, skills distilled from trajectories let GPT-5.6 Sol suggest more diagnostic and actionable interventions.

are messy signals containing low-level operations such as updating an implementation, changing the training configuration, or switching to another dataset. ResearchTrials contain higher-level abstractions about research decisions on methods, experiments, and ablations. This gives us a view of how a research project evolves before arriving at the final discoveries. Furthermore, we abstract project-agnostic research patterns from trajectory samples and observe insightful research strategies. We study how ResearchTrials can be useful for improving AI-assisted scientific exploration through multiple use cases, including test-time and training-time methods. First, we use the trajectories as external knowledge in the form of demonstration data or skills. Second, we explore supervised and reinforcement fine-tuning on research trajectories to explore the next research decision. We showcase that ResearchTrails enables models to learn from the sequence of scientific exploration steps.

Our contributions are summarized as follows. In Section 3, we introduce ResearchTrails and a scalable pipeline for extracting evidence-grounded research-decision trajectories from Git repositories. In Section 4, we show that these trajectories capture meaningful patterns of scientific exploration, including how methods, experiments, and ablations evolve over time. In Section 5, we demonstrate that ResearchTrails can support scientific exploration as test-time knowledge through demonstrations and reusable skills, and provide training-time supervision through ordered research-decision sequences.

2 Related Work

Benchmarks for autonomous research. Recent work has developed benchmarks for evaluating LLMs on different stages of scientific research, such as idea generation (Si et al., 2025; Qiu et al., 2025), hypothesis generation and evaluation (Liu et al., 2025b;a; O'Neill et al., 2025), experimentation (Huang et al., 2023; Chan et al., 2025a), paper writing (Wang et al., 2024), or paper reviewing (Liu & Shah, 2023; Idahl & Ahmadi, 2025). Other benchmarks explore multiple tasks in broader research workflows, including executing concrete scientific tasks (Chen et al., 2025; Du et al., 2026; Zhao et al., 2026; Starace et al., 2025) and evaluating AI agents on open-ended machine learning research (Chen et al., 2026a; Nathani et al., 2025). These benchmarks increasingly capture realistic components of scientific research, but they primarily evaluate task completion or quality based on the final output rather than how research decisions evolve over the course of a project.

Methods and systems for research agents. Beyond benchmarking, recent systems have developed new tools, algorithms, and pipelines to automate the general research process. For example, Shao et al. (2025) build agentic systems that integrate web browsing, document retrieval, and reasoning to produce answers and reports. Other work improves different part of the research process such as idea generation (Baek et al., 2025; Li et al., 2025) or using scientific skills (Kassis et al., 2026; K-Dense Inc., 2026; Zhang, 2026). Extensive works have built end-to-end research automation pipelines or

simulated research communities (Lu et al., 2024; Yamada et al., 2025; Schmidgall et al., 2025; Tang et al., 2025; Gottweis et al., 2025; Yu et al., 2024). Despite advancements of these approaches, their research trajectories are largely generated by LLMs or induced by predefined agentic workflows, rather than learned from the development histories of human researchers.

Human research artifacts. Research papers, though often only record the final research outcome, have been widely used to provide training signals for research agents (Goel et al., 2025; He-Yueya et al., 2026; Liu et al., 2026). Several recent AI scientist systems explicitly incorporate humans into the research loop, allowing humans to provide feedback for LLMs (Gottweis et al., 2026; Swanson et al., 2025). Concurrent work (Chen et al., 2026b) distills files in the GitHub repositories into LLM skills, whereas we focus on specific repositories with rich development histories and study their commit histories as evidence of how human research decisions evolve.

Learning from process traces. A growing line of work uses intermediate process traces to provide supervision for various tasks. ScholaWrite (Le et al., 2025) uses LaTeX edits to improve AI writing, and SWE-RL (Wei et al., 2026) mines pull requests from open-source software to help reproduce developers' patches. In contrast, we focus on scientific exploration, and we aggregate many low-level commits into research decisions about methods, experiments, and ablations.

3 RESEARCHTRAILS: FROM GIT HISTORIES TO RESEARCH TRAJECTORIES

In this section, we present the formulation of research decision trajectories, introduce how such data can be extracted from existing GitHub repositories at scale, and present our workflow in detail.

We consider the *research exploration* problem where we expect a large language model (LLM) system to assist with insightful discussion on how to proceed with the existing research project. Given a research query on what has been attempted on a research question, the AI-assisted workflow suggests the next research decision that is promising. Specifically, it can propose a change in the project development, such as 'add load balancing loss to the routing objective', 'evaluate model performance after being $10\times$ deeper', or 'visualize the loss landscape'.

The desirable data for this task is the process of human research explorations, which unfortunately is typically not presented in a paper. However, our insight is that the GitHub repository associated with a paper can maintain historical commits that reflect a long period of explorations. Individual commits can be too low-level to represent meaningful research decisions, whereas a sequence of commits and code differences can collectively correspond to higher-level research decisions. For example, the commit sequence

change router implementation, update the training bash script, visualize experiment results

yields the decision 'study new router as an alternative to the original routing method'.

Therefore, we extract a research decision sequence $(a_1, a_2, \ldots, a_T)$ (a research trajectory) from the commit sequence. In this work, we focus on repository-observable research decisions that can be systematically grounded in repository evidence. We formulate each a_t as a scientific change that meaningfully alters a project's proposed method, experimental design, or ablation study.

Multiple commits may be grouped into one decision when they correspond to the same underlying scientific change, and one commit can reflect multiple decisions. In addition, we separate research decisions from ordinary engineering changes: formatting changes, dependency updates, typo corrections, and file differences from model weights or notebook metadata are generally not treated as research decisions.

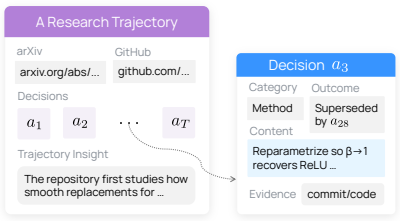


Figure 2: Structure of a research trajectory (left) and a research decision (right).

Each research decision is represented in a data structure containing textual descriptions of decision `content`, together with commit ID references and code evidence that supports the annotation (Figure 2). In addition, each decision has two property labels describing its `category` and `outcome`. A `category` label is either *method*, which builds or revises the project's contribution itself; *experiment*, which changes how the contribution is evaluated; or *ablation*, which studies the sensitivity of the components of the proposed method. An `outcome` label is

either *retained*, which remains in effect at the end of the observed history; *superseded*, which a later decision replaces in the same research role; or *abandoned*, which is removed without a clear successor. We present the instructions for labeling decisions in Appendix A.

We present our workflow for automatically extracting decisions from commit histories in Figure 3, which is composed of four steps: (1) discover the paper/repository pairs for a given vanue-year and filter out pairs whose repository has no progressive commits; (2) collect code differences in commits as evidence excerpts; (3) extract decisions from code evidences, and (4) annotate the decisions following the requested data structure. The details of each step are described in Section 3.1–3.3, with further implementation designs in Appendix A.

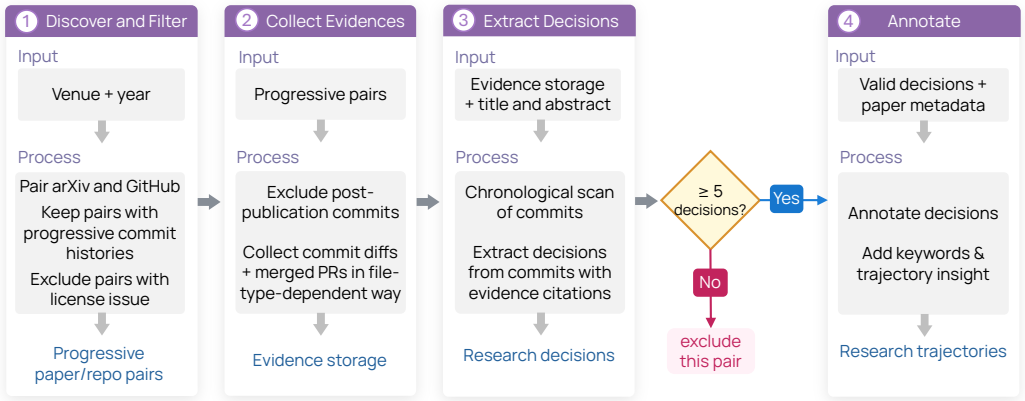


Figure 3: Workflow of extracting research decisions from GitHub repositories. Candidate paper/repository pairs are discovered and filtered before collecting repository code evidence. An LLM then processes that evidence chronologically, merges commits into decisions, with citations checked against the stored Git evidence. Only repositories with at least five validated decisions are annotated.

3.1 Selecting repositories with observable research development

In this stage, given a venue and year, we retrieve the paper list from Semantic Scholar (Kinney et al., 2023) and retain paper/repository pairs with progressive commit histories before entering the next steps. First, we resolve and pair the arXiv and GitHub links: For each paper with an arXiv identifier, we collect all the GitHub repository links from the latest arXiv PDF text, excluding References, Bibliography, or Appendix, so that links appearing only in the supplementary material are not harvested. Next, a candidate repository is retained only if its README contains the paper's full title. We then count the number of commits before the arXiv date and record the number of days between the first commit and the arXiv date. We exclude paper/repo pairs with fewer than 5 pre-publication commits or fewer than 30 span days between the first commit date and paper publication date.

We additionally apply the following filtering rules: We exclude known repositories that maintain several papers' commits in one repo; we exclude papers with fewer than 2 pre-publication commit messages that match a pool of research-related keywords; we exclude papers whose GitHub license restricts code usage for research purposes. Pairs surviving every filtering rule are labeled *progressive*.

3.2 Evidence-grounded trajectory extraction

In the extraction stage, we turn a commit history into a research trajectory under two principles. First, one decision may take many commits to implement, and one commit may implement several decisions. Their mapping is *many-to-many*, and we present our method to scan over commits below. Second, we enforce *annotation reliability* by grounding every decision with repository evidence in the form of code pieces.

Evidence storage. For each progressive pair, we create a partial repository clone that omits file contents larger than a specified size threshold, and walk its default branch up to the end of the paper's

arXiv upload month. Commits are enumerated following the commit time. Where the GitHub API associates merged pull requests (PR) with default-branch commits, the PR commits are interleaved at their merge anchors so that work developed on a branch appears at the point it entered the history.

We record commit contents into an 'evidence storage' according to the following rule: (1) Informative file types such as source, configuration, and notebook files are *read at creation*. The first commit that adds a path stores the file's complete text up to a character cap, and every later commit changing that file records only a bounded diff. (2) Notebooks are parsed to retain only the code input, removing large spurious diffs produced by simply re-running a notebook. (3) Artifacts such as model checkpoints and logs only have metadata stored, such as path and change status.

Mapping commits to decisions. A *decision* expresses exactly one scientific choice regarding the *method*, *experiment*, or *ablation*. Setup code, loaders, logging, and reporting utilities are merged into the choice they implement. In this step, we call LLM APIs to extract decisions from commit histories. LLMs read the stored evidence file together with the title and abstract of each paper only, which are references to assist with determining, for example, whether a commit is research-relevant or a side-product of a library upgrade.

Each LLM call receives the earlier decisions before the current batch of commits, based on which it classifies the role of each commit: *irrelevant*, *support* (adds evidence to an existing decision), *revise* (changes one decision), *new* (begins one), *multiple* (contains several), or *uncertain*. An uncertain commit is stored in an evidence-backed *open thread*. An open thread may later be merged into an existing decision, promoted to being a decision together with other commits, or rejected if never merged or promoted. An example of mapping a commit to decisions is presented in Table 2.

Grounding decision annotations with repository evidence. If a commit is classified as supporting one or multiple decisions, its ID will be added to those decisions' evidence list. Then the LLM response contains assessments of every commit in the batch, created or revised decisions, created or revised open threads, thread resolutions, and decision merges. If the response omits any commit's assessment or names a decision unsupported by evidence, we discard the entire response and retry the commit under a halved input budget. After obtaining the commit assessments, the harness applies this delta to the complete decision trajectory, unioning commit identifiers and evidences.

3.3 Verification and trajectory annotation

After all commits have been accessed, the LLM validates every claimed commit identifier, file path, date, and code difference excerpt against the evidence storage. An excerpt that is not an exact substring of a stored patch is discarded, and a decision with no valid evidence citation is rejected. Each decision records its lifecycle outcome (*retained*, *abandoned*, or *superseded*), where a superseded decision must name its successor; a successor link whose target does not survive validation is dropped.

Accepted decisions are ordered by their first cited commit date. A repository with fewer than five validated decisions is dropped. Otherwise, the LLM performs one final call to derive keywords and *trajectory insight*, a trajectory-level observation from the validated decisions alone.

4 Dataset Properties and Research Trajectory Patterns

In the following, we discuss the dataset properties in Section 4.1 and present several general research patterns extracted from the annotated research trajectories in Section 4.2.

4.1 Dataset Overview

Utilizing the workflow in Section 3, we process 15K papers from NeurIPS 2022–2025 due to this venue's relatively broad topic coverage and abundant indices of arXiv links on Semantic Scholar. We find 5K out of the 15K indexed papers have a matched repository. Next, the filtering stage retains 677 progressive pairs (13%), and the final dataset contains **599** paper/repository pairs with annotated human research trajectories, which consist of **13,271** research decisions. This showcases that out of all the collected papers, $\sim$4% of them can yield a progressive research decision trajectory.

Stage	Outcome	Count			
		2022	2023	2024	2025
Input	Papers indexed for the venue/year	2790	3587	4544	4313
Stage 1	No arXiv identifier	558	693	862	934
	No repository link found	1142	1477	1779	1670
	README does not name the paper	214	237	330	321
	Retained pairs	**845**	**1198**	**1594**	**1400**
Stage 2	Fewer than 5 total commits	190	235	333	340
	Fewer than 5 pre-paper commits	279	420	499	394
	Pre-paper span below 30 days	187	282	413	380
	Fewer than 2 research commits	72	92	116	112
	Progressive pairs	**111**	**168**	**229**	**169**
Stage 3	Stored commits	16,661	25,345	43,664	41,646
	Stored evidences	104,981	173,469	2,759,132	514,430
Stage 4	Fewer than five decisions	16	12	22	5
	Annotated pairs	**90**	**150**	**204**	**155**
	Annotated decisions	**2176**	**3593**	**5260**	**2242**

Table 1: Count of paper/repository pairs throughout the workflow for NeurIPS 2022–2025.

Table 3 in the appendix reports the distribution of commit accesses, where 57% of commits are *irrelevant*: In repositories showing meaningful commit activities before the publication date, roughly one commit in three carries a research decision.

For our annotated 13,271 decisions, they comprise 5226 changes in methods (39%), 6826 in experiments (51%), and 1219 in ablations (9%). Among these, 76.3% are retained, 11.5% are abandoned, and 12.1% are superseded. Decision counts per trajectory range from 5 to 205, with a median of 15, and 35% of all the trajectories have ≥ 20 decisions. A decision has 7.5 evidence citations on average.

4.2 How research trajectories evolve

A research trajectory and its trajectory insight are project-specific. To show what carries over between projects, we abstract recurring decision sequences into *research patterns* and draw each as a chain of the current state connecting the next decisions. Figure 4 shows four research patterns, from which we make two observations. First, the next step depends on the project's state more than on its topic—$a_{<t}$ describes a project status without being specific to a domain, and the pattern has instances in trajectories in wide domains. Since patterns are generalizable across projects, they can be given to a language model as guidance or skills for choosing a next decision, which we examine in Section 5. Second, the patterns investigate why progress stalled before changing the method, so the skills can prompt a diagnosis rather than a quick list of all potential ways to change the proposed method.

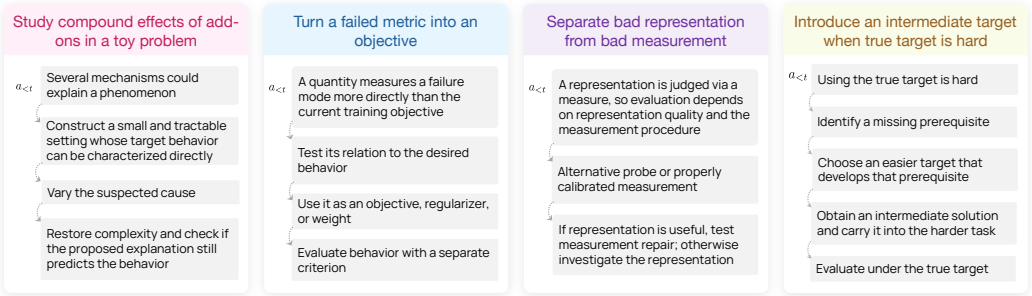


Figure 4: Research patterns extracted by GPT-6 Astra.

We provide examples of our annotated trajectories in Figures 11–14 in Appendix A, which display winding research processes with back-and-forth development of methods, experiments, and ablations.

5 APPLICATIONS

Given an instruction prompt p and the previous human research decisions $a_{<t} = (a_1, \ldots, a_{t-1})$, we denote the input to LLM as $h_t = (p, a_{<t})$. Our goal is to obtain an improved agentic workflow that can assist with human research exploration, i.e., yielding a promising next decision $\hat{a}_t \sim \pi_\theta(\cdot|h_t)$ that leads to observations that expand the boundaries of human knowledge.

Since it remains an open question to evaluate the effectiveness of an arbitrary decision $\hat{a}_t$ (Si et al., 2026a; Garikaparthi et al., 2026), in this work, we propose using held-out real human research trajectories that we collected as user simulations, and we measure if the generated next decision is strategically similar to the actual decision. We also present case studies showing the improved quality of the generated next decision. We use an LLM judge (Zheng et al., 2023; Liu et al., 2023) to evaluate the similarity between two research decisions via scoring three aspects of a research decision independently with {0,1,2}: *component match* (what part of the project is acted on, such as baselines, evaluation datasets, model architecture, etc.), *operation match* (the scientific action, such as introduce, extend, replace, remove, extend, etc.), and *specification match* (concrete choices). We then define `score` as the sum of the match in *component* and *operation*, discarding *specification* since, for example, 'increase batch-size to 16' and 'increase batch-size to 32' are strategically the same research decision. We present the judge rubric in Appendix B and examples in Table 5. We use judge GPT-5.6 Sol, whose judge score variance and judge quality are evaluated in Appendix B.

In the following, we propose both test-time and training-time approaches to improve decision similarity scores on held-out human research trajectories. For test-time approaches, we derive harness by (1) using a few trajectories as demonstration data or (2) distilling skills from trajectories, which are compared with (3) distilling skills from final papers only. For the training-time approach, we train an autoregressive policy over the next decision $\pi_\theta(a_t|h_t)$ via supervised and reinforced fine-tuning.

5.1 HARNESS: USE TRAJECTORY EXAMPLES AS DEMONSTRATIONS AND SKILLS

To understand the difficulty of predicting the next decision, we vary the number of prior decisions provided in context and compare the scores obtained by predictor Gemini 3.1 Flash-Lite. We observe in Figure 5 that performance saturates around 3–5 recent decisions, indicating that the next decision becomes less uncertain with abundant decision history, such as at a late stage of a project.

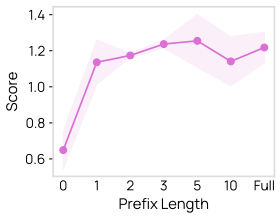


Figure 5: Prefix information plateau.

Method	Prefix-1	Prefix-2	Prefix-3
Baseline	1.256 ± .090	1.289 ± .022	1.445 ± .063
All demos	1.328 ± .095	1.316 ± .068	1.373 ± .071
Random two demos	1.241 ± .037	1.216 ± .020	1.274 ± .121
Retrieve two demos	1.361 ± .043	1.460 ± .016	1.498 ± .080
Shuffled skill	1.363 ± .045	1.308 ± .182	1.391 ± .053
Final paper skill	1.500 ± .020	1.231 ± .064	1.271 ± .034
Skills	**1.697** ± .088	**1.557** ± .089	**1.607** ± .009

Figure 6: Decision similarity of the predicted next decision based on previous $\{1, 2, 3\}$ decisions. Judge scores are averaged across evaluation trajectories, each with 3 runs.

To focus on the most challenging scenario where decision history is missing and the next decisions are uncertain, we predict the next decision using prefixes in early timesteps of lengths $\{1, 2, 3\}$. We utilize 19 training repositories' trajectories for training and 134 held-out repositories for evaluation. We use Gemini 3.1 Flash Lite to predict the next decision based on the previous trajectory and compare the following methods: (1) **Baseline:** no harness; (2) **All demos:** use all the training trajectories as demonstrations (Brown et al., 2020); (3) **Random two demos:** randomly choose two training trajectories as demos; (4) **Retrieve two demos:** predictor selects two most relevant training trajectories as demos; (5) **Skill:** use skills distilled from the training trajectories by Opus 5; (6) **Shuffled skill:** use skills distilled from the same trajectories with the temporal order randomly permuted within each project; (7) **Final paper skill:** use skills distilled from the final paper content only.

We observe that experience from other projects effectively helps predict the next decision when a project has only 1-3 decisions so far. The two most effective methods are 'Retrieve two demos' and

'Skills': Retrieval outperforms random selection by 0.196; Skills perform the best, outperforming retrieval by 0.181, final paper skills by 0.286, and baseline by 0.290 on average. For a qualitative study, we present the generated next decisions and their scores on held-out real trajectories in Figure 7. Furthermore, we run user simulations where a researcher asks GPT-5.6-Sol research queries (Figure 8), observing that skills let Sol suggest smaller and more interpretable interventions.

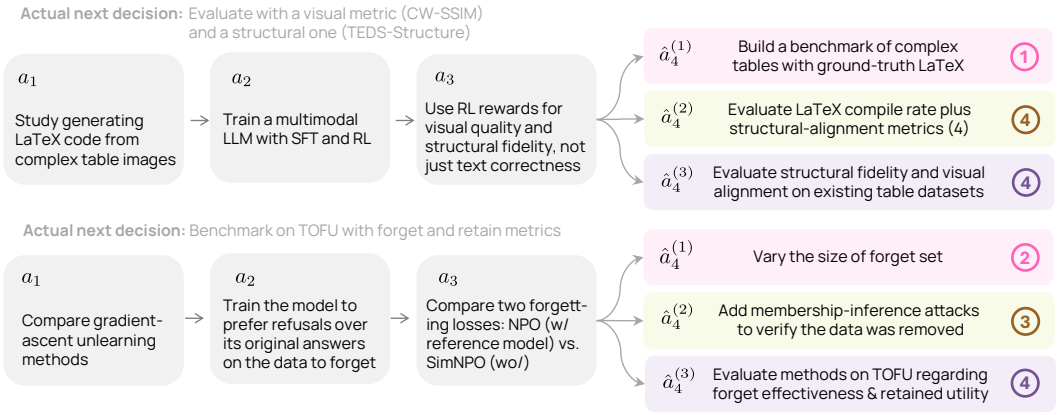


Figure 7: Sampling three next decisions based on previous decisions from held-out real trajectories. Upper: The decisions are generated with method 'Skills'; Bottom: The decisions are generated with 'Retrieve two demos'. The circled values after generated decisions are the decision similarity scores.

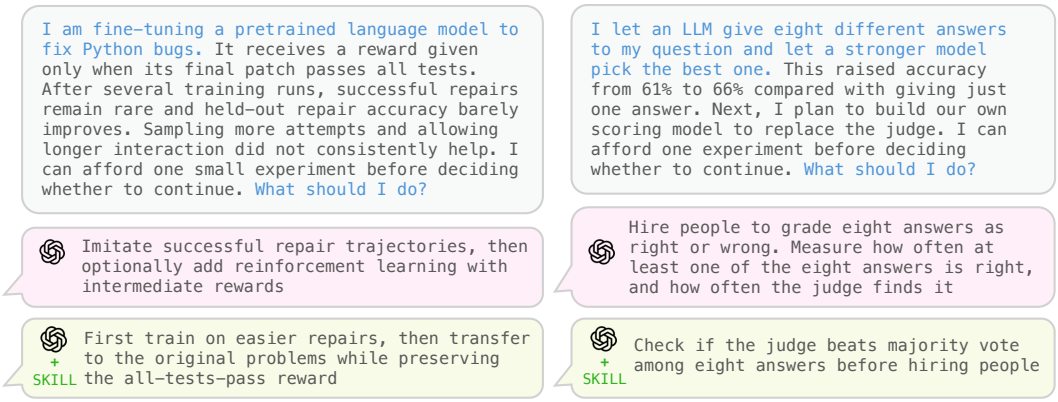


Figure 8: User simulations where a user asks GPT-5.6 Sol (high reasoning) research queries. In both examples, the skills let Sol suggest smaller and more interpretable interventions, changing the suggestion from **trying a remedy** to **determining which component needs repair**.

5.2 Training: Autoregressive Decision Modeling and RL

For the training-time approach, we present a two-stage pipeline: autoregressive decision modeling (SFT cold start), followed by GRPO (Shao et al., 2024) with decision similarity rewards. In the SFT stage, we fine-tune $\pi_\theta(a_t|h_t)$, a Qwen3-8B model (Yang et al., 2025), to predict the next decision via

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(p,a_{1:T})\sim\mathcal{D}_{\text{train}}}\left[\sum_{t=1}^{T}\log \pi_\theta\left(a_t|p, a_{<t}\right)\right].$$

where a_t is a sequence of tokens describing the decision content. Suppose it has length $|a_t| = S$, then $\log \pi_\theta(a_t|p, a_{<t}) = \sum_{s=1}^{S} \log \pi_\theta(a_{t,s}|p, a_{<t})$. This objective learns the distribution $p(a_{1:T}|p)$, which represents plausible ways that a human research project may evolve given an initial instruction.

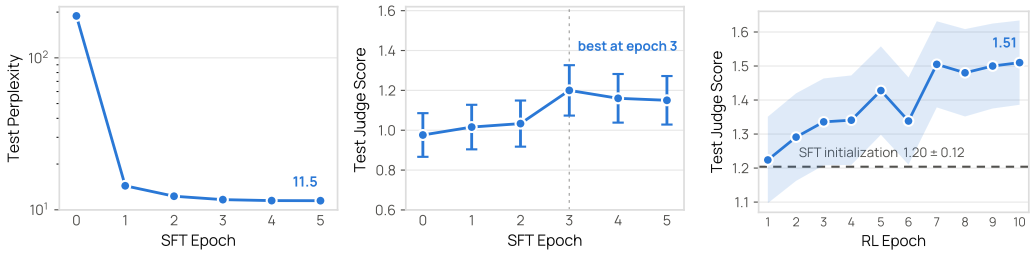


Figure 9: Test performance of two-stage training, including SFT cold-start (left and middle) and RL with next-decision rewards (right). We observe that SFT improves the probability of the next decision (left) whereas the generation quality plateaus at epoch 3, and the RL stage further improves generation quality (right). The error bars and shades indicate the score's 95% confidence interval.

We observe in Figure 10 (left and middle) that likelihood and generation quality diverge during SFT. Test perplexity continuously falls, whereas the judge score of greedy predictions peaks at epoch 3 and does not improve further. Further gains therefore require optimizing generation quality directly.

Therefore, during RL, we use the SFT checkpoint as the frozen reference policy π_{ref}. For each trajectory prefix, we sample K decisions $\{\hat{a}_k\}_{k=1}^{K}$ from the current policy. The decision similarity judge assigns the component c_k and operation match o_k against a, yielding reward $R_k = c_k + o_k$ if $\hat{a}_k$ is a valid generation, and 0 otherwise. We optimize the policy with Dr. GRPO (Liu et al., 2025c), observing that RL effectively further improves the judge score from 1.20 to 1.51 (Figure 10, right).

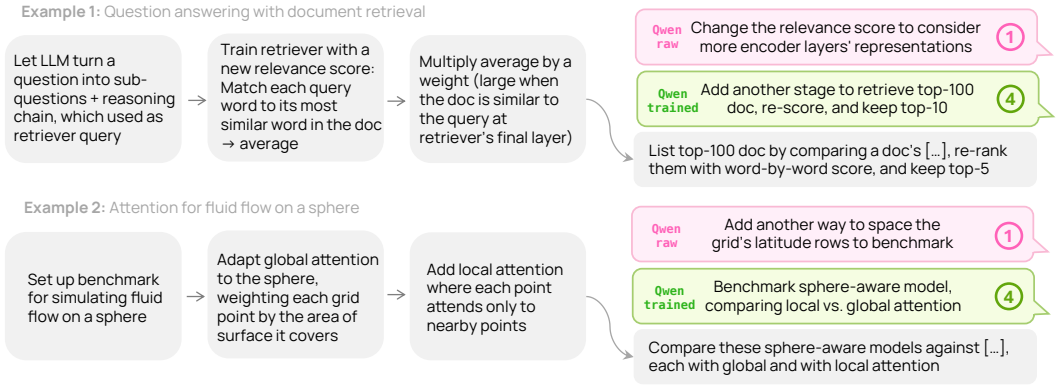


Figure 10: Generated next decisions from Qwen3-8B before and after our two-stage training. The examples are from held-out real trajectories, with grey blocks indicating the annotated decisions. The circled values are decision similarity scores for the corresponding generated decisions.

6 Conclusion and Future Work

We introduced ResearchTrails, a dataset of human research decision trajectories extracted from the GitHub commit histories. These trajectories record what final papers omit, such as old method versions, abandoned experiments, and ablations that shaped the final contribution. We observe and distill generalizable research skills from our trajectories. At test time, skills distilled from our trajectories improve frontier models' capability of generating the next decision on held-out projects, outperforming skills distilled from the final papers alone. At training time, we fine-tune Qwen3-8B via SFT and reinforcement learning with decision-similarity rewards, showing that our workflow is a scalable approach for providing learning supervision for scientific exploration.

We consider two directions for future work. First, we have focused on artificial intelligence conferences, while code histories can be maintained in other scientific research areas such as biology and physics. Second, our dataset only contains projects with Git commits, while there are other platforms containing research artifacts that can reflect research exploration patterns.

AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing writing, dataset curation (cleaning and reformatting the dataset), and interpreting results in LLM-as-judge. We have not used generative AI tools for proposing/refining hypotheses or providing feedback on research methodology. Other disclosure tasks, including assisting with translation or mathematical proving, are not applicable to this work. We have reviewed all AI-assisted work by checking the dataset curation quality. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide visualizations of the annotated data examples in the appendix. The appendix documents the experimental setup, including dataset construction, model configurations, hyperparameters, and evaluation protocols. Our implementation, data processing scripts, training and evaluation code, and experimental configurations will be released upon acceptance.

REFERENCES

Akari Asai, Jacqueline He, Rulin Shao, Weijia Shi, Amanpreet Singh, Joseph Chee Chang, Kyle Lo, Luca Soldaini, Sergey Feldman, Mike D'arcy, et al. Openscholar: Synthesizing scientific literature with retrieval-augmented lms. *arXiv preprint arXiv:2411.14199*, 2024.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6709–6738, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ziyang Cai and Harkirat Behl. Ai scientist via synthetic task scaling. *arXiv preprint arXiv:2603.17216*, 2026.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025a.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, et al. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, volume 2025, pp. 50466–50494, 2025b.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. Mlr-bench: Evaluating ai agents on open-ended machine learning research. *Advances in Neural Information Processing Systems*, 38, 2026a.

Jianlyu Chen, Yuyang Hu, Hongjin Qian, Jiawei Liu, Wenqing Wei, Xiaolong Chen, Defu Lian, Zhicheng Dou, Chaozhuo Li, Qiwei Ye, et al. Repo-to-skill: Distilling github repositories into ai4ai skills. *arXiv preprint arXiv:2609.02749*, 2026b.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, volume 2025, pp. 96934–96990, 2025.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Licheng Zhang, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents. In *International Conference on Learning Representations*, volume 2026, pp. 42414–42448, 2026.

Kevin Niall Dunbar. How scientists really reason: Scientific reasoning in real-world laboratories. *The nature of insight*, 1994.

Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76 (5):378, 1971.

Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. Researchgym: Evaluating language model agents on real-world ai research. *arXiv preprint arXiv:2602.15112*, 2026.

Shashwat Goel, Rishi Hazra, Dulhan Jayalath, Timon Willi, Parag Jain, William F Shen, Ilias Leontiadis, Francesco Barbieri, Yoram Bachrach, Jonas Geiping, et al. Training ai co-scientists using rubric rewards. *arXiv preprint arXiv:2512.23707*, 2025.

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

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with co-scientist. *Nature*, pp. 1–3, 2026.

Joy He-Yueya, Anikait Singh, Ge Gao, Michael Y Li, Sherry Yang, Chelsea Finn, Emma Brunskill, and Noah D Goodman. Giants: Generative insight anticipation from scientific literature. *arXiv preprint arXiv:2604.09793*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Leyang Hu, Matteo Gamba, and Randall Balestriero. Curvature tuning: Provable training-free model steering from a single parameter. *Advances in Neural Information Processing Systems*, 38: 60938–60979, 2026.

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

Maximilian Idahl and Zahra Ahmadi. Openreviewer: A specialized large language model for generating critical scientific paper reviews. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pp. 550–562, 2025.

K-Dense Inc. Claude scientific skills: A comprehensive collection of scientific tools for claude ai. `https://github.com/K-Dense-AI/claude-scientific-skills`, 2026. skills covering databases, packages, integrations, and analysis tools.

Timothy Kassis, Vinayak Agarwal, Yuhuan He, Darshil Patel, and Aubrey M Brueckner. Scientific agent skills: A library of procedural knowledge for research agents. *arXiv preprint arXiv:2609.00065*, 2026.

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

David Klahr and Kevin Dunbar. Dual space search during scientific reasoning. *Cognitive science*, 12 (1):1–48, 1988.

Patrick Tser Jern Kon, Qiuyi Ding, Jiachen Liu, Xinyi Zhu, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, et al. Exp-bench: Can ai conduct ai research experiments? In *International Conference on Learning Representations*, volume 2026, pp. 120555–120604, 2026.

Khanh Chi Le, Linghe Wang, Minhwa Lee, Ross Volkov, Luan Tuyen Chau, and Dongyeop Kang. Scholawrite: A dataset of end-to-end scholarly writing process. *arXiv preprint arXiv:2502.02904*, 2025.

Long Li, Weiwen Xu, Jiayan Guo, Ruochen Zhao, Xingxuan Li, Yuqian Yuan, Boqiang Zhang, Yuming Jiang, Yifei Xin, Ronghao Dang, et al. Chain of ideas: Revolutionizing research via novel idea development with llm agents. In *EMNLP (Findings)*, pp. 8971–9004, 2025.

Haokun Liu, Sicong Huang, Jingyu Hu, Yangqiaoyu Zhou, and Chenhao Tan. Hypobench: Towards systematic and principled benchmarking for hypothesis generation. *arXiv preprint arXiv:2504.11524*, 2025a.

Jiachen Liu, Maestro Harmon, and Zechen Zhang. Sci-reasoning: A dataset decoding ai innovation patterns. *arXiv preprint arXiv:2601.04577*, 2026.

Ryan Liu and Nihar B Shah. Reviewergpt? an exploratory study on using large language models for paper reviewing. *arXiv preprint arXiv:2306.00622*, 2023.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 2511–2522, 2023.

Yujie Liu, Zonglin Yang, Tong Xie, Jinjie Ni, Ben Gao, Yuqiang Li, Shixiang Tang, Wanli Ouyang, Erik Cambria, and Dongzhan Zhou. Researchbench: Benchmarking llms in scientific discovery via inspiration-based task decomposition. *arXiv preprint arXiv:2503.21248*, 2025b.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, et al. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

Charles O'Neill, Tirthankar Ghosal, Roberta Răileanu, Mike Walmsley, Thang Bui, Kevin Schawinski, and Ioana Ciucă. Sparks of science: Hypothesis generation using structured paper data. *arXiv preprint arXiv:2504.12976*, 2025.

Yansheng Qiu, Haoquan Zhang, Zhaopan Xu, Ming Li, Diping Song, Zheng Wang, and Kaipeng Zhang. Ai idea bench 2025: Ai research idea generation benchmark. *arXiv preprint arXiv:2504.14191*, 2025.

Robert Rosenthal. The file drawer problem and tolerance for null results. *Psychological bulletin*, 86 (3):638, 1979.

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

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. In *International Conference on Learning Representations*, volume 2025, pp. 94003–94092, 2025.

Chenglei Si, Tatsunori Hashimoto, and Diyi Yang. The ideation-execution gap: Execution outcomes of llm-generated versus human research ideas. In *International conference on learning representations*, volume 2026, pp. 74195–74240, 2026a.

Chenglei Si, Zitong Yang, Yejin Choi, Emmanuel Candès, Diyi Yang, and Tatsunori Hashimoto. Towards execution-grounded automated ai research. *arXiv preprint arXiv:2601.14525*, 2026b.

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. Language agents achieve superhuman synthesis of scientific knowledge. *arXiv preprent arXiv:2409.13740*, 2024. URL https://doi.org/10.48550/arXiv.2409.13740.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: evaluating ai's ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

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

Hoai-Chau Tran, Duy Nguyen, Duy M Nguyen, TrungTin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Zou, Binh T Nguyen, and Mathias Niepert. Accelerating transformers with spectrum-preserving token merging. *Advances in Neural Information Processing Systems*, 37: 30772–30810, 2024.

Heng Wang, Pengcheng Jiang, Jiashuo Sun, Zhiyi Shi, Haofei Yu, Jiawei Han, and Heng Ji. Learning to predict future-aligned research proposals with language models. *arXiv preprint arXiv:2603.27146*, 2026.

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

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Advances in Neural Information Processing Systems*, 38:78500–78525, 2026.

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

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

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

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

Yongjun Zhang. Vibe researching as wolf coming: Can AI agents with skills replace or augment social scientists?, 2026. arXiv:2602.22401.

Bingchen Zhao, Despoina Magka, Minqi Jiang, Xian Li, Roberta Raileanu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Kelvin Niu, Shagun Sodhani, Michael Shvartsman, et al. The automated llm speedrunning benchmark: Reproducing nanogpt improvements. *Advances in Neural Information Processing Systems*, 38, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Dataset Details

In the following, we further detail the design considerations and implementations of our workflow for extracting research decisions from GitHub repositories. We use GPT-5.6 Sol (xhigh reasoning) for annotating decisions and writing trajectory insights.

Handling repository forks. A repository fork inherits commits its authors did not write, and those commits should not be counted as the paper's research history. However, some type of fork is an exception: a personal repository forked into a laboratory organization after the work is complete has a creation timestamp that postdates every research commit, and windowing by that timestamp would discard the entire genuine history. We therefore distinguish the two cases: if the *parent's* README also contains the paper's full title, the parent is the authors' earlier repository and the whole history is theirs; otherwise the parent is unrelated upstream, and every subsequent measurement counts only commits at or after the fork's creation timestamp.

Evidence storage. We record commit contents into an 'evidence storage' by the following rule:

- Source, configuration, and notebook files are *read at creation*. The first commit that adds a path stores the file's complete text up to a character cap, which depends on the file type: 50K for Python, other source, and notebooks; 20K for shell and configuration. Every later commit changing that file records only a bounded diff.
- Notebooks are parsed to retain only each cell's type and source (i.e., code input), which removes the large spurious diffs in cell outputs produced by simply re-running a notebook.
- Binary artifacts, model checkpoints, logs, and files under the output directories carry only metadata, including path, change status, size, and the reason the contents were withheld.

Batching. Commits are grouped into units—one per commit, subdivided when a single commit's evidence exceeds a character budget—and packed greedily into requests until either the model's usable input context or the estimated output budget for the batch would be exceeded. In our descriptions, we abuse the word 'commit' for 'unit'.

Commit assessment example. Suppose one commit introduces a new loss function, adds an independent evaluation comparison, and reformats a utility file. The intended output contains content in Table 2.

Output	What it records
Commit assessment	`multiple`, referencing a_1 and a_2
Decision a_1	The loss-function choice, citing its relevant code
Decision a_2	The evaluation choice, citing its relevant code

Table 2: Example of mapping a commit to decisions, where we suppose one commit introduces a new loss function, adds an independent evaluation comparison, and reformats a utility file.

Assessment	Meaning	Count	Percentage
`irrelevant`	no research decision	66,712	56.6%
`support`	evidence for an existing one	17,640	15.0%
`multiple`	several decisions at once	16,544	14.0%
`revise`	changes an existing one	8,130	6.9%
`new`	begins a new one	5,130	4.4%
`uncertain`	held as an open thread	3,696	3.1%
Total		**117,852**	

Table 3: Commit assessments for 599 annotated pairs for NeurIPS 2022–2025.

Cost and throughput. Regarding frontier LLM usage, the cost is input-dominated: input tokens to output tokens has a ratio 16:1. Table 4 shows that 99% of LLM cost is in `commit_scan`, the only step that transmits repository evidence.

Function	Calls	Input Token / Call	Output Token / Call	Cost Percentage
`commit_scan`	8,543	143,042	9,175	99.0%
`commit_finalize`	708	18,918	1,353	0.5%
`annotation`	671	16,409	528	0.4%
Total	**9,922**			**100%**

Table 4: LLM cost by stage for NeurIPS 2022–2025.

Prompt for annotation

```
You are analyzing a repository chronologically to recover research decisions observable
in its public history.

PAPER CONTEXT
The paper title and abstract are supplied only to determine whether a repository change
concerns the linked paper. They are not evidence for a research decision.
Do not reconstruct decisions, motivations, results, or methods from the paper.

RESEARCH DECISION
A research decision is a repository change that meaningfully affects the paper's
proposed contribution, how that contribution is evaluated, or the settings and
components of the proposed method.

Routine formatting, documentation, CI, dependency updates, bug fixes, and refactoring
are not research decisions unless repository evidence shows that they alter the
scientific method or evaluation.

CATEGORY
Give every decision exactly one category: `method`, `experiment`, or `ablation`. Label
what the decision changes, not why it was made; wording such as "Hypothesize",
"Investigate", or "Ablate" does not decide the category. Use the paper title and
abstract only to identify the paper's contribution: what the paper builds or proposes
(a model, objective, algorithm, estimator, theory, training or inference procedure, or
system), or, for a benchmark, dataset, simulator, or platform paper, that released
artifact. For an empirical-study or analysis paper, the contribution is the analysis,
and the models it trains are experimental setup.
- `method`: the decision builds or revises the contribution itself.
  - This covers its architecture, representation, objective or loss form, training or
  inference procedure, estimator, data pipeline, theory, and claims about its scope.
  - For a system, it covers the system's operators and internals.
  - For a benchmark, it covers the benchmark's data, tasks, ground-truth targets,
  splits, environments, and construction rules.
  - Introducing a mechanism with its initial settings, replacing one mechanism with a
  different one, and creating a knob that is the paper's key novelty are all method.
- `experiment`: the decision changes how the contribution is evaluated or run for
evaluation.
  - This covers evaluation datasets and conditions (data size, noise, dimension,
  domain), metrics and scoring, protocols, checkpoint selection, seeds, budgets, and
  reporting.
  - It covers baselines and comparators, including their hyperparameters. In a
  benchmark paper, the models run on the benchmark are baselines.
  - It covers training settings of the models an analysis paper studies, analyses and
  diagnostics, and comparisons against baselines.
- `ablation`: the decision changes the value of a hyperparameter of the proposed
method, when that hyperparameter is not itself the paper's key novelty.
  - Hyperparameters include the training settings of the authors' own model: learning
  rate, batch size, epochs, optimizer or schedule, dropout, EMA, loss weights,
  temperatures, thresholds, sizes, and counts.
  - This holds whether the value is changed once, edited back and forth, or swept, and
  whether or not side-by-side runs are visible.
  - A controlled comparison among variants of the proposed method's own components (A
  vs B, with vs without) is also ablation.
```

Prompt for annotation (cont.)

```
- Key-novelty knobs. This rule applies when the hyperparameter is itself the paper's
key novelty. It also applies to a weight, strength, or level that controls the proposed
term or mechanism, such that some value of it turns the method back into an existing
baseline (for example, a weight of 0 that removes the proposed term). Creating the knob
is `method`. A later change or sweep of its value is:
  - `experiment` if any value tried recovers an existing baseline, including when that
  value is one of several swept values;
  - `ablation` if no value tried does;
  - `experiment` if the evidence does not show whether any value recovers a baseline.
- Proposed metrics. When the contribution is itself a metric, score, evaluation method,
or verifier, building it is `method`. Adopting or configuring standard metrics is
`experiment`.
- Analysis papers. The paper's studied factor is `experiment`. A side-by-side
comparison of the training recipe of the models it studies is `ablation`.
- Tie-break. A decision that both builds a mechanism and evaluates it takes the
category of its main content:
  - `method` when that is building or revising the mechanism;
  - `ablation` when it is a controlled comparison of the method's own variants;
  - `experiment` when it is a comparison against baselines or an evaluation setting.

PROCESS
1. Process the supplied commits strictly in sequence order. Assess every commit. Do not
silently omit one.
2. A commit may:
    - be research-irrelevant;
    - support an existing decision;
    - revise an existing decision;
    - begin a new decision;
    - combine several earlier commits into one decision; or
    - contain several distinct decisions when separate changes support them.
3. Merge engineering commits implementing the same scientific choice into one decision.
A merge is transitive: if C was merged with A and later D supports C, D supports the
existing {A,C} decision unless new evidence explicitly shows that the earlier grouping
was wrong. Do not silently discard A or an intervening candidate B merely because C or
D matched another thread.
4. Preserve uncertain or incomplete changes exclusively as evidence-backed entries in
`open_threads` so later commits can clarify them. Never put an uncertain candidate in
`decisions` with an open status. For example, if an engineering implementation that
itself is not a decision but it could be a decision if later commits implement the
remaining, that engineering implementation can be an open thread. Every new or updated
thread must include the relevant commit SHAs and exact repository citations from the
current call; the harness retains its earlier evidence locally. Uncertainty means the
evidence is not yet sufficient to classify the scientific choice, not that the thread
lacks evidence.
5. Never invent intent, causality, failed experiments, or observations.
6. Cite only supplied repository evidence. A diff excerpt must be copied exactly.
7. Preserve existing decision IDs. When merging decisions, record which IDs were merged
rather than silently deleting them.
   `merge_log` may name only formal decision IDs supplied in `prior_state`. If several
   changes in the current call belong to one atomic decision, return that one decision
   directly; do not invent and immediately absorb temporary decision IDs.
8. A prior open thread may be resolved in three ways:
    - If it belongs to an existing prior decision, merge its commit SHAs and evidence
    into that decision, remove the thread, and record `merged_into_decision` with the
    existing decision ID.
    - If it becomes a distinct supported decision, remove the thread and record
    `promoted_to_decision` with its decision ID.
    - If later evidence shows that it is not a research decision, remove the thread and
    record `rejected` with an empty result decision ID.
9. The response is a delta, so omit unchanged prior open threads from
`open_thread_upserts`; the harness retains them locally. Emit exactly one
`thread_resolutions` entry only when the current call resolves and removes a prior
thread. `thread_resolutions` contains only resolutions made in the current call; it is
not a cumulative history.

DECISION VALIDITY AND OUTCOME
`status` records whether a candidate is genuinely a research decision. `outcome`
records what later happened to a valid decision. Do not use one as a substitute for the
other.
- Use `status: "supported"` for every genuine historical research decision, including
choices that were later removed, disabled, or replaced.
- Use `status: "rejected"` only to remove an existing decision candidate that later
evidence or final consolidation shows was never a research decision, such as setup-only
work or an implementation step belonging to a parent choice. A rejected candidate must
have `outcome: null`. Do not create a rejected decision for a newly observed irrelevant
change, and use `merge_log` rather than rejection for duplicates.
- Use `outcome: "retained"` when the decision remains operative at the end of the
observed history or no supplied evidence shows that it was abandoned or superseded.
```

Prompt for annotation (cont.)

```
- Use `outcome: "superseded"` only when a later extracted, supported decision clearly
performs the same research role and repository evidence shows that it replaces the
earlier choice. Preserve both decisions and set `superseded_by` to the successor's
decision ID. A related later experiment, implementation change, deletion, or inactive
invocation is not enough by itself.
- Use `outcome: "abandoned"` for every other discontinued decision, including one that
is removed or disabled without such a clearly identified successor. When uncertain
whether a later decision is a true replacement, use `abandoned`.
- Set `superseded_by` to null for every outcome other than `superseded`, and for every
rejected candidate. Name only a decision ID that exists in this response or in
`prior_state`; never invent one.
- A deletion, disabled invocation, or replacement can update a supported decision's
outcome and add lifecycle evidence. It must not by itself change that decision's status
to rejected.
- When a merge combines duplicate decision records, set the result's outcome from the
latest lifecycle evidence. Upsert the merge target when its outcome changes.

RECENT-FIRST MATCHING
For each new research-relevant commit, compare it first with open threads in
`recent_first_matching_order.open_thread_ids`, then with established decisions in
`recent_first_matching_order.decision_ids`. Stop at the first clearly compatible
scientific choice. Recency is a search preference, not a requirement: a nonadjacent
commit may join an older thread or decision when the semantic evidence is clear.
Intervening unmatched threads remain open and independent. Reject a thread only when
evidence shows that it is not a research decision or it remains unsupported during
finalization. If evidence instead establishes that the thread was a genuine decision
later abandoned or superseded, promote it with that outcome and, when superseded, with
its successor's decision ID. Never reject it merely because another thread received a
match.

ATOMIC DECISION GRANULARITY
Each decision must express exactly one independently meaningful scientific choice,
comparison, or revision. Apply these tests consistently:
- Merge code changes, configuration edits, loaders, scripts, logging, and reporting
utilities when they only implement or support the same method or experimental
comparison.
- Keep two choices separate only when each independently changes the proposed method, a
dataset comparison, the evaluation protocol, or an ablation. In this case, the
successor formulation counts as an independent decision.
- A change in mathematical form, objective, architecture, training mechanism, inference
procedure, or scientific data flow is a new decision even when it replaces--and cannot
coexist with--the prior decision.
- A sweep over a single parameter, distribution, schedule, or component is one
decision, however many commits set individual values; its category follows the CATEGORY
rules. Record the swept range in the decision text rather than splitting it.
- The same rule applies across components: applying one baseline or normalization
scheme to a second model, branch, or stage is support for the existing decision unless
it introduces an independent research comparison of its own, e.g., augmenting the
experiment evaluation protocol.
- A new dataset or model is a separate decision only when repository evidence shows
that it introduces a distinct research comparison, not merely support code.
- A fallback, bug fix, parameter-counting utility, or result parser is not a separate
decision unless it intentionally changes the scientific data, metric, protocol, or
interpretation.
- Use one consistent level of scope within a benchmark. Do not combine many attack or
model families into one decision while treating a comparable single integration as a
separate decision without a scientific reason.
- Do not inflate the decision count by turning hyperparameters or implementation steps
of exactly one scientific choice into separate decisions.

If a current decision becomes too broad, keep its independently variable choices
separate while their full evidence is available. During finalization, merge over-split
decisions and reject setup-only entries using the same tests. Preserve valid historical
decisions with abandoned or superseded outcomes. Record every merge in `merge_log`.

INPUT
- paper_context: title and abstract
- prior_state: previously accumulated decisions and open threads
- recent_first_matching_order: open-thread and decision IDs ordered from most recently
evidenced to oldest
- commits: the next chronological commit batch
- evidence records: selected nonempty textual evidence only, with evidence ID, SHA,
date, path, change status, evidence kind, and retained content. A creation snapshot is
the bounded file text from its first add event; later records for that path contain
only diffs. Empty and manifest-only records remain in local evidence.json and are not
supplied.
```

Prompt for annotation (cont.)

```
RETURN JSON
The response is a delta, not the complete rolling state.
- `decision_upserts` contains only decisions created, supported, semantically revised,
assigned a new outcome, or explicitly rejected as invalid in this call. For an existing
decision, return its complete updated semantic fields but only commit SHAs and exact
citations newly supplied in this call. Omit unchanged prior decisions.
- `open_thread_upserts` contains only threads created or updated in this call, with
only newly supplied commit SHAs and exact citations. Omit unchanged prior threads.
- `thread_resolutions` and `merge_log` contain only operations performed in this call.
- A `thread_resolutions` entry may name only an ID listed in
`resolvable_open_thread_ids`. When that list is empty, return an empty
`thread_resolutions` array.
- The harness retains and unions all omitted prior decisions, threads, commit SHAs, and
citations locally.

{
"commit_assessments": [
    {
    "commit_sha": "...",
    "classification": "irrelevant|support|revise|new|multiple|uncertain",
    "decision_ids": ["Dxxx"],
    "reason": "..."
    }
],
"decision_upserts": [
    {
    "decision_id": "Dxxx",
    "decision": "...",
    "category": "method|experiment|ablation",
    "status": "supported|rejected",
    "outcome": "retained|abandoned|superseded or null when rejected",
    "superseded_by": "decision ID of the successor, or null unless outcome is
    superseded",
    "commit_shas": ["..."],
    "evidence": [
        {
        "commit_sha": "...",
        "date": "...",
        "path": "...",
        "diff_excerpt": "..."
        }
    ],
    "why_research_relevant": "..."
    }
],
"open_thread_upserts": [
    {
    "thread_id": "Txxx",
    "summary": "...",
    "commit_shas": ["..."],
    "evidence": [
        {
        "commit_sha": "...",
        "date": "...",
        "path": "...",
        "diff_excerpt": "..."
        }
    ],
    "needed_evidence": "..."
    }
],
"thread_resolutions": [
    {
    "thread_id": "Txxx",
    "resolution": "merged_into_decision|promoted_to_decision|rejected",
    "result_decision_id": "Dxxx or empty when rejected",
    "reason": "..."
    }
],
"merge_log": [],
"batch_limitations": []
}

After the final batch, code verifies every SHA, path, and exact diff excerpt;
consolidates duplicates; removes rejected candidates; retains supported decisions of
every outcome; drops a `superseded_by` link whose successor did not survive validation;
orders decisions by their first cited commit date; and derives decision_count itself.
The annotation LLM then receives only those validated decisions and their evidence, not
the entire accumulated conversation.
```

Judge rubrics

```
You receive (1) an observable trajectory of research decisions, (2) the actual next
decision, and (3) one prediction for the next decision. Evaluate how closely the
prediction matches the actual decision.

Score `component_match` (what part of the project is acted on), `operation_match` (the
scientific action), and `specification_match` (concrete choices) independently from 0
to 2.

## Understand the decision before scoring

Describe each decision as an **operation on a component, with specifications**. Use the
trajectory to resolve references and understand whether a choice is being introduced,
extended, replaced, or tested.

| Decision | Component | Operation | Specifications |
| --- | --- | --- | --- |
| Add CIFAR10 to the existing evaluation suite. | Evaluation dataset selection | Extend
the suite | CIFAR10 |
| Replace the current loss with a weighted sum of classification and reconstruction
losses. | Training objective | Replace the objective | The two loss terms and their
weighted combination |
| Compare training with and without the reconstruction loss. | Training objective |
Ablate a loss term | Reconstruction loss |

Identify the functional component; treat named implementations, such as AdamW versus
SGD, as specifications.

## 1. `component_match`

Does the prediction act on the same functional part of the project as the actual
decision?

**0 -- A different part of the project, or no identifiable component.**

- One changes the optimizer, while the other changes training-data construction.
- One changes the model architecture, while the other changes the loss objective.

**1 -- A related part, or only a broader identification of the correct part.**

- One changes an evaluation metric, while the other changes how uncertainty in that
metric is estimated. Both concern measurement, but act on different parts of it.
- One changes token embeddings, while the other changes positional encodings. They
concern related representation components.
- The actual decision changes the evaluation metric, but the prediction only says
"improve evaluation," without identifying a particular component.

**2 -- The same functional component.**

- Both change the optimizer, even if they choose different optimizers.
- Both act on a hyperparameter search grid, even if one expands it and the other
restricts it.

Tuning an optimizer and replacing it concern the same component. Identify experiments
by the functional part they change, not their experiment numbers or dataset names.

## 2. `operation_match`

Does the prediction perform the same scientific action? Examples include introduce,
replace, remove, extend, restrict, compare, ablate, calibrate, and validate. For
questions and hypotheses, identify whether the statement asks a question, proposes an
explanation, or tests it.

**0 -- A different or opposing operation, or no identifiable action.**

- One expands the hyperparameter search range, while the other narrows it.
- One adds a neural-network layer, while the other moves an existing layer to a
different location.
- One proposes an explicit ablation, while the other merely says "improve the method"
without identifying an action.

**1 -- A related but distinct operation, or an underspecified version of the action.**

- One introduces a new tunable hyperparameter, while the other extends the search grid
for an existing hyperparameter.
- One replaces the optimizer with a candidate optimizer, while the other benchmarks
that candidate against the current optimizer before making a choice.
- The actual decision replaces a mechanism, while the prediction modifies that
mechanism.
```

Judge rubrics (cont.)

```
**2 -- The same operation, including semantic paraphrases.**

- One introduces a baseline, while the other introduces a metric: both introduce,
despite acting on different components.
- "Ablate the auxiliary loss" and "compare training with and without the auxiliary
loss" both describe an ablation.

## 3. `specification_match`

Do the concrete choices agree, either exactly or at the functional-family level allowed
by this rubric? Specifications include datasets, model families, architectures, loss
terms, numeric thresholds, split ratios, and hyperparameter values. This dimension
intentionally allows broader matches than exact names alone.

**0 -- No matching or meaningfully comparable concrete settings.**

- CIFAR10 versus ReLU: an evaluation dataset versus an activation function.
- Setting batch size to 32 versus 128: the component and operation match, but the
values do not.
- The actual decision names AdamW, while the prediction says only "replace the
optimizer": no replacement optimizer is specified.

**1 -- Partial agreement, or related settings used for different tasks.**

- CIFAR10 versus COCO object detection: both involve image data, but the tasks differ.
- The learning rate matches, but batch size differs or is omitted.

**2 -- The defining settings agree.**

- CIFAR10 versus CIFAR100 when used as image-classification evaluation datasets.
- Rotten Tomatoes versus SST-2 when used as sentiment-classification evaluation
datasets.
- ResNet versus ViT when the choice is a generic image-feature backbone and the
architecture's internal mechanism is not the subject of the decision.

Apply family equivalence to corresponding roles, not to any shared technical term.
ResNet and ViT can match as generic backbones, but not when convolution versus
self-attention is itself being investigated.

If the actual decision supplies no concrete specifications, assign specification score
0 without lowering the other scores.

## Worked comparisons across all three dimensions

| Actual decision | Predicted decision | Component | Operation | Specification |
| -- | -- | -- | -- | -- |
| Replace the optimizer with AdamW. | Switch the training optimizer to AdamW. | 2 | 2 |
2 |
| Replace the optimizer with AdamW. | Replace the optimizer. | 2 | 2 | 0 |
| Replace the optimizer with AdamW. | Compare AdamW against the current optimizer
before selecting one. | 2 | 1 | 2 |
| Replace the optimizer with AdamW. | Replace the tokenizer with BPE. | 0 | 2 | 0 |
| Ablate the auxiliary loss. | Permanently remove the auxiliary loss to simplify
training. | 2 | 1 | 0 |
| Set batch size to 32. | Set batch size to 128. | 2 | 2 | 0 |
| Increase batch size to 128. | Decrease batch size to 32. | 2 | 0 | 0 |
| Set batch size to 32 and learning rate to 0.001. | Set batch size to 128 and learning
rate to 0.001. | 2 | 2 | 1 |
| Add CIFAR10 to the evaluation suite. | Add COCO object detection to the evaluation
suite. | 2 | 2 | 1 |
| Introduce a new evaluation metric. | Introduce an uncertainty estimate for the
existing metric. | 1 | 2 | 0 |

## Scoring discipline and output

- Score the semantic content of the `decision` sentence. If predicted `category`,
`object`, or `operation` fields are present, use them only to clarify ambiguity. They
cannot override the sentence or supply an action that it does not state. Predictions
with only `category` and `decision` can be scored normally.
- Do not penalize component or operation scores for missing or different specification.
They can receive 2 even when specification match is 0. Conversely, matching concrete
names does not establish matching components or operations. Identify each dimension
separately before assigning its score.
- Judge meaning rather than writing quality.
- If a prediction contains several independent alternatives, do not select whichever
one matches best. Judge what it actually commits to in each dimension.
```

Actual (A) and predicted (P) decisions	C	O	S
A: Replace the optimizer with AdamW. P: Switch the training optimizer to AdamW.	2	2	2
A: Replace the optimizer with AdamW. P: Replace the optimizer.	2	2	0
A: Replace the optimizer with AdamW. P: Compare AdamW against the current optimizer before selecting one.	2	1	2
A: Replace the optimizer with AdamW. P: Replace the tokenizer with BPE.	0	2	0
A: Ablate the auxiliary loss. P: Permanently remove the auxiliary loss to simplify training.	2	1	0
A: Set batch size to 32. P: Set batch size to 128.	2	2	0
A: Increase batch size to 128. P: Decrease batch size to 32.	2	0	0
A: Set batch size to 32 and learning rate to 0.001. P: Set batch size to 128 and learning rate to 0.001.	2	2	1
A: Add CIFAR10 to the evaluation suite. P: Add COCO object detection to the evaluation suite.	2	2	1
A: Introduce a new evaluation metric. P: Introduce an uncertainty estimate for the existing metric.	1	2	0

Table 5: Examples of comparing research decisions across three dimensions. 'C' for *Component*, 'O' for *Operation*, and 'S' for *Specification*.

B Application Details

B.1 Test-Time Harness

Judge variance and quality. We experiment with judge GPT-5.6 Sol, observing that judge scores of fixed inputs have consistently small variance < 0.001 (Table 6). Further, we evaluate judge quality by comparing judge scores with ground-truth scores: We synthesize paraphrases of the actual decisions, adjacent decisions to the actual ones, and unrelated decisions to the actual ones via varying *Component*, *Operation*, and *Specification*. We observe consistent matching between the obtained and expected scores (Table 7).

We judge the same predicted decisions using GPT-5.6 Sol for three times, observing that

Metric	Sample SD	Fleiss' Kappa (Fleiss, 1971)
`Score`	0.021	0.888
Component	0.006	0.935
Operation	0.015	0.912

Table 6: Judge variance measured with predictions fixed.

Control	Expected `Score`	`Score`	*Component*	*Operation*
Same	4	4.00	2.00	2.00
Paraphrase	4	4.00	2.00	2.00
Adjacent	2	2.39	1.34	1.05
Unrelated	0	0.10	0.04	0.06

Table 7: Judge's scores match the expected scores when evaluating the actual decisions, their faithful paraphrases, adjacent decisions, and unrelated decisions.

Skills

```
# Early-Stage Next-Decision Prediction

Use this when the full observable history is one, two, or three decisions and the
project is genuinely young. Nothing earlier is hidden; the prefix is the whole project
so far. Your job is to name **one** next decision: its category (`method`, `experiment`,
`ablation`), the **object** acted on, and the **operation**.

## Step 1 -- Build the inventory from the prefix

Read the prefix and mark which of these exist:

- **Mechanism**: a proposed procedure, objective, loss, estimator, formulation, or
model component.
- **Data**: a dataset, corpus, benchmark, or simulation/data-generation setup.
- **Measurement**: a metric, score, judge, diagnostic, or scoring protocol.
- **Tunables**: any parameter, threshold, temperature, radius, granularity, rate,
count, or on/off switch explicitly named.
- **Revision signal**: a prefix item marked as superseded by a later observed item, or
wording like "replace", "instead of", "reparameterize".

Almost every observed next decision acts on an object already present in the prefix or
on the immediately adjacent missing slot of the same pipeline. Do **not** predict a new
unrelated object, a new task domain, a theory/analysis section, or a baseline
comparison with outside work unless the prefix already names it.

## Step 2 -- Apply gap rules in priority order

1. **Data present, measurement absent → predict `experiment`: define how outcomes are
scored.** This is the most reliable transition. Typical form: an evaluation protocol,
error/quality metric, model-judge scoring, conditional-independence or correlation
diagnostic, or a pass/fail criterion applied to the existing data. Also fires when the
first decision is a mechanism plus a chosen corpus but no metric.

2. **Data and measurement both present, mechanism absent → predict `method`: introduce
the project's core mechanism** (objective, estimator, training procedure) or the
data-construction/labeling pipeline that feeds it. Prefixes that open with an
evaluation suite, pilot comparison, or baseline comparison already containing metrics
move to method next.

3. **Mechanism present, no data and no measurement → choose between two
continuations.** Predict another `method` decision when the prefix mechanism looks
provisional or partial: a coarse prompt-based heuristic, a degenerate parameter trick,
one stage of an obviously multi-stage system, or a component with adjacent uncovered
stages (target construction, labeling, loss term, efficiency, resolution/granularity
choice). The usual operation is *replace with a finer-grained variant of the same
stage* or *add the next stage*. Predict an `experiment` setting up data/metrics instead
when the mechanism reads as complete and self-contained (a full detector, formulation,
or loss with nothing obviously missing).

4. **Mechanism just introduced with one salient continuous knob or discrete
granularity, and no evaluation exists yet → predict `ablation`: sweep that knob.**
Prefer this over rule 3 when the prefix foregrounds a parameter range, a
smoothness/scale factor, or a resolution choice as the mechanism's defining degree of
freedom.

5. **Mechanism plus measurement present, no ablation yet → predict `ablation` over the
newest named tunable or the most recently introduced component.** Early ablations are
always sweeps of, or on/off toggles of, something already written in the prefix (a
threshold, a scale, a prompt variant, a sample population, a structural alternative),
never novel machinery.

6. **Prefix ends on a narrow synthetic/toy experiment or ablation → predict
`experiment` that widens scope**: more conditions, a broader or differently spaced
parameter range, more datasets from the same family, or the same measurement
transferred from the synthetic setting to a real pretrained/real-data setting.

7. **Revision signal present.** If a method stage has already been superseded and no
evaluation of it exists, predict yet another `method` revision of that same stage. If
an ablation or experiment already targeted the pre-revision version, predict re-running
or extending that same ablation/experiment under the revised version.

8. **All-method prefixes of length two or three with no data or metric anywhere**
continue in `method`, advancing along the pipeline (construction → granularity/scope →
objective → estimation/efficiency/validation). Switch to `experiment` only when every
core stage the prefix implies is already specified; then predict the dataset/condition
configuration.
```

Skills (cont.)

```
9. **All-experiment prefixes** continue in `experiment`, alternating between
data/condition generation and diagnostics/metrics on those conditions, until both are
covered.

## Step 3 -- Fallback for ambiguous prefixes

If two rules conflict or none clearly fires, predict the **same category as the most
recent decision, acting on the same object, as an incremental extension or replacement
of it**. This is the modal behavior at all three prefix lengths. Prefer `method` over
`experiment` when nothing in the prefix has yet been measured; prefer `experiment` when
data exists but nothing has been scored.

## What cannot be inferred this early

State predictions at the object-and-operation level. Do **not** commit to:

- numeric sweep values, thresholds, learning rates, sample counts, seeds, or split
boundaries;
- the identity of specific external models, tokenizers, or judge models;
- the full list of datasets in a later evaluation suite (name the family implied by the
prefix instead);
- whether the next decision will be retained, superseded, or reverted;
- long-run architecture choices, theory results, or final comparisons.

When rule 5 or 6 fires, name the swept knob taken verbatim from the prefix vocabulary
and say the range is underdetermined. When rule 1 fires, name the measurement type
suggested by the prefix's object (e.g., ranking/threshold quality for detection-style
objects, reconstruction/forecast error for generative or emulation objects, correctness
scoring for question-answering objects) rather than inventing a specific metric
formula.

## Output form

One sentence naming the category, then one or two sentences naming the object plus
operation, then one clause stating the chief alternative you rejected and which
inventory gap decided it. Keep specifications generic wherever the prefix does not pin
them down.
```

B.2 Two-Stage Training

Denote W_0 as the pretrained weights, and the weights after SFT are $W_{\text{SFT}} = W_0 + \Delta W_{\text{SFT}}$. We initialize a policy π_θ by merging SFT LoRA into the model weights and then attaching a fresh LoRA adapter. Therefore, we have $W_\theta = W_{\text{SFT}} + \frac{\alpha}{r} B_\theta A_\theta$. We freeze W_{SFT}, initialize A_θ randomly and B_θ to zero, and optimize only the new LoRA weights. We freeze the reference policy π_{ref} with weights W_{SFT}.

Let $\mathcal{D}_{\text{train}} = \{(h_i, a_i)\}_{i=1}^N$ contain research trajectory prefixes h_i and their annotated next decisions a_i. Each prefix includes the fixed instruction and the preceding decisions. A generated decision a is a sequence in JSON format with `category` and `content` fields. After SFT finishes, we use the resulting checkpoint as the frozen reference policy π_{ref}. The reward is

$$R_{ik} = \begin{cases} c_{ik} + o_{ik}, & \text{if } \hat{a}_{ik} \text{ is valid JSON and ends with EOS,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We retain a group only when $s_i > 0$, which is computed by

$$\overline{R}_i = \frac{1}{K} \sum_{k=1}^{K} R_{ik}, \qquad \widehat{A}_{ik} = R_{ik} - \overline{R}_i, \qquad s_i = \sqrt{\frac{1}{K} \sum_{k=1}^{K} (R_{ik} - \overline{R}_i)^2}. \tag{2}$$

Denote $p_{ikt}^{\Theta} = \pi_{\Theta}^{\tau}(y_{ik,t} | h_i, y_{ik,<t})$ as the probability of a sampled token under the current policy ($\Theta = \theta$), the policy that generated the batch ($\Theta = \text{old}$), or the SFT reference ($\Theta = \text{ref}$). We take a single update per batch, so GRPO's ratio clipping is inactive and omitted. With Dr. GRPO normalization Liu et al. (2025c) and the GRPO reference penalty Shao et al. (2024), the loss is

$$\mathcal{L}_{\text{GRPO}} = \frac{1}{|\mathcal{S}| K C} \sum_{i \in \mathcal{S}} \sum_{k=1}^{K} \sum_{t=1}^{T_{ik}} \left[-\widehat{A}_{ik} \frac{p_{ikt}^{\theta}}{p_{ikt}^{\text{old}}} + \beta \left(\frac{p_{ikt}^{\text{ref}}}{p_{ikt}^{\theta}} - \log \frac{p_{ikt}^{\text{ref}}}{p_{ikt}^{\theta}} - 1 \right) \right], \tag{3}$$

where $\mathcal{S}$ is the set of prefixes in the batch with non-constant rewards, K is the group size, and T_{ik} is the number of tokens in the generated decision k.

Algorithm 1 SFT Cold Start and Reinforcement Learning with Decision Similarity Rewards

Require: $\mathcal{D}_{\text{train}}$, pretrained policy π_θ, group size K

// SFT Stage

Train π_θ on $\mathcal{D}_{\text{train}}$ by minimizing $\mathcal{L}_{\text{SFT}}$

Freeze copy $\pi_{\text{ref}} \leftarrow \pi_\theta$

// RL Stage

for each batch $\mathcal{S}$ **do**

    Sample K decisions from $\pi_\theta^\tau(\cdot \mid h_i)$ for all $i \in \mathcal{S}$

    Compute rewards R_{ik} via Eq. (1) and $\mathcal{L}_{\text{GRPO}}$ via Eq. (3)

    Update θ with $\nabla_\theta \mathcal{L}_{\text{GRPO}}$

end for

return π_θ

Implementation. We perform SFT for 5 epochs with batch size 16 and select the best checkpoint as the initialization for RL training. We perform RL updates for 10 epochs using batch size 8, $K = 8$, $\beta = 0.04$, $\epsilon = 0.2$, $C = 64$, and $\tau = 0.8$. Our LoRA (Hu et al., 2021) uses rank $r = 8$ and $\alpha = 16$. We use a training dataset with 125 trajectories at with beginning prefix lengths {1,2,3}, yielding 375 training samples. We use the AdamW optimizer Loshchilov & Hutter (2017) with a constant learning rate 10^{-4}, no weight decay, and gradient norm clipping at 1. We use GPT-5.6 Sol with medium reasoning as the judge.



Figure 11: Annotation example for Hu et al. (2026). The annotation extracts methodological and experimental research decisions by reading the commit histories of its GitHub repository. Dates indicate the first repository evidence for each decision.

Trajectory Summary

The repository first investigates how smooth replacements for ReLU affect what a network learns. Early attempt introduces SmoothReLU and trains toy networks at different smoothness. The method later shifts to post-hoc steering: train or load a trained ReLU model, replace ReLUs with the proposed activations, and vary the β scalar across all replaced modules. The activation itself evolves from a sigmoid-gated form to BetaAgg, a combination of sigmoid-gated and Softplus branches with a default coefficient c=0.5.

Evaluation expands from toy problems to in-domain image classification, cross-dataset transfer, adversarial and corruption robustness, semantic segmentation, and additional ResNet and Swin backbones. For in-domain and robustness experiments, replacement can be entirely post-hoc with pretrained weights fixed. For transfer and segmentation, a new task-specific head still needs to be fitted; the repository compares fitting that head before vs. after replacement and later favors replacement followed by head fitting.

Ablations examine c values, beta values, base-model capabilities, noisy data, feature depth, and backbone families. Unproductive explorations including noisy-training, transferred-robustness, whole-network segmentation training, and VGG are abandoned.

By the v1 cutoff, Curvature Tuning has become a method for steering a pretrained backbone; it is training-free only when the existing output head can be reused, whereas transfer to a new task still requires fitting a new head.

Figure 12: Annotation example for Hu et al. (2026) (cont.).



Figure 13: Annotation example for Tran et al. (2024). The annotation extracts methodological and experimental research decisions by reading the commit histories of its GitHub repository. Dates indicate the first repository evidence for each decision.

T19 (Method, 20 Mar 2024)
Remove to size and attention penalties
For image layers with m < 0.45, use mean ELU scores and remove the added size/attention penalties in T18.
1 evidence, superseded by T21

T20 (Experiment, 30 Mar 2024)
Evaluate on visual QA
Merge visual tokens in LLaVA CLIP image encoder. Vary token-retention rates and compare answer quality and runtime with other token-compression methods.
32 evidences

T21 (Method, 3 Apr 2024)
Score neighbors with Gaussian weights
Give nearby tokens more weight than distant tokens, then average weights:
$$E_i = \frac{1}{N}\sum_{j=1}^{N} \exp\left[-\left(\frac{s_{ij}}{1-m}\right)^2\right]$$
6 evidences

T22 (Method, 4 Apr 2024)
Text: return to raw similarity
Replace the transformed text score with sum plus mean of cosine similarities.
2 evidences, superseded/return to T21

T23 (Method, 9 Apr 2024)
Text: delete the most redundant tokens
Delete r text tokens with the highest Gaussian redundancy score, without transferring the deleted tokens' features
3 evidences, superseded by T30

T24 (Method, 11 Apr 2024)
Images: try one fixed Gaussian width
Set σ = 0.5 instead of 1 - m, and stop multiplying scores by $1/(\sigma\sqrt{(2\pi)})$.
1 evidence, superseded/return to T21

T25 (Method, 3 May 2024)
Images: alternate sources and receivers
Among ranked tokens, use odd-rank ones as sources and even ones as receivers. Each source merges into its most similar receiver.
3 evidences

T26 (Method, 8 May 2024)
Reverse how the cutoff sets width
Try σ = m instead of σ = 1 − m, i.e., increasing m widens the Gaussian neighborhood rather than narrows it.
2 evidences, superseded/return to T21

T27 (Ablation, 8 May 2024)
Compare with attention-based methods
On BLIP image–caption retrieval, compare PiToMe's accuracy and runtime with two attention-based selection variants at matching token-retention. Also compare with BLIP without token reduction.
4 evidences

T28 (Ablation, 8 May 2024)
Compare fixed vs. adaptive widths
At 95% token retention, compare 9 fixed Gaussian widths in 0.1~2.0 against adaptive σ = 1 − m in image–caption retrieval accuracy and compute cost.
5 evidences

T29 (Method, 9 May 2024)
Images: Use ToMe at larger cutoffs
When m ≥ 0.45: split ordinary token order into alternating source/receiver groups and merge the r strongest source matches. Use Gaussian candidate selection for smaller m.
1 evidence

T30 (Method, 13 May 2024)
Text: combine token features again
Restore merging: split 2r ranked tokens into r sources followed by r receivers; each source chooses its most similar receiver and contributes feature to receiver's average.
1 evidence

Trajectory Summary

The project starts with ToMe's established accuracy–speed benchmark, asking how to identify redundant tokens before merging them. Early PiToMe scores reward similarity above a cutoff and penalize dissimilar pairs. The code repeatedly changes this rule: hard thresholds give way to smooth ELU scores, temporarily return to raw cosine similarity, and later become Gaussian neighbor scores. Image and text models follow partly different paths.

Choosing candidates is only one part of the method. The history also changes which tokens may receive merged information, how sources and receivers are divided, and whether selected features are averaged or discarded. A broad receiver pool returns to a restricted pool; text pruning later returns to merging. Gaussian-width trials return to the adaptive rule σ = 1 − m. The later experiments compare similarity-based selection with attention-based selection and compare fixed with adaptive Gaussian widths. The retained implementation combines Gaussian scoring with a separate ToMe-style rule at larger cutoffs.

Trajectory insight: preserving useful information requires coordinating three choices—what counts as redundancy, where information can be merged, and how much compression is applied.

Figure 14: Annotation example for Tran et al. (2024) (cont.).