TRAINING LLMS TO GUIDE RESEARCH BY STUDYING SCIENTIFIC PAPERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training language models to make progress on open-ended research problems is difficult because useful feedback is hard to attribute to individual research decisions, such as proposing an idea, developing a method, or designing an experiment. Directly specifying rewards for such intermediate decisions can also become overly prescriptive, narrowing what counts as a promising research direction. In AI research, even obtaining sufficient reliable feedback is quite expensive. We introduce PESTO, or "Paper-Extracted Surrogate Task Optimization," an approach for improving a model's ability to conduct open-ended AI research by deriving focused research tasks from scientific papers. We decompose papers into connected *decisions* such as identifying literature gaps, proposing methods, designing experiments, and interpreting results, and turn them into shorter-horizon problems with criterion-level rubric rewards. Applying this construction across many papers makes individual decisions easier to evaluate while preserving a broad range of research approaches. We train a 27B-parameter model with RL on tasks derived from ICML, ICLR, NeurIPS, and arXiv papers. Training improves performance on held-out research tasks and transfers to more open-ended settings where proposed ideas need to be implemented and evaluated. PESTO improves downstream research outcomes and shows qualitative evidence of broader exploration. In a case study, it develops an extension that improves upon the reproduced baseline of an ICML 2026 oral paper excluded from our training corpus.

1 INTRODUCTION

Training with reinforcement learning (RL) has made large language models (LLMs) increasingly capable of solving hard problems in domains such as math and coding. A natural next step is to train models to make progress on more open-ended research problems, such as AI research, where the model must decide what problems to pursue, what assumptions to make, what experiments to run (Lu et al., 2024), and how to use the resulting evidence to decide what to try next. In domains such as coding, learning from feedback over long traces works well because feedback is repeatable and can often be localized to concrete failures. Feedback for open-ended research problems is less straightforward. Evaluating a single idea may require substantial implementation and experimentation, outcomes can be noisy, and a failed experiment does not necessarily imply that the underlying research direction or even the design choice was poor. Obtaining enough reliable execution feedback to distinguish among many possible research directions can therefore become prohibitively expensive, particularly when individual research traces span multiple days.

An alternative is to provide direct supervision on individual segments of a long research trajectory, but this raises a different problem: what should count as a good next step? Hand-designing detailed rewards or rubrics over a research process lasting one or two weeks requires specifying what promising research should look like. This is particularly difficult in open-ended research, where several different approaches may be worth pursuing and their value is often unknown until they are tried (Si et al., 2026). If the supervision becomes too prescriptive, RL may concentrate the model on approaches that match these criteria while discouraging other useful directions (Chen et al., 2026a; Lanchantin et al., 2025a). We therefore need a source of scalable supervision for individual research decisions that does not require executing every proposed idea or prescribing a single research path in advance.

To address this, we introduce a training phase built from focused research-decision tasks extracted from scientific papers. Our key observation is that a completed paper connects many of the same decisions a model must make when tackling an open-ended research problem: identifying a gap,

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.

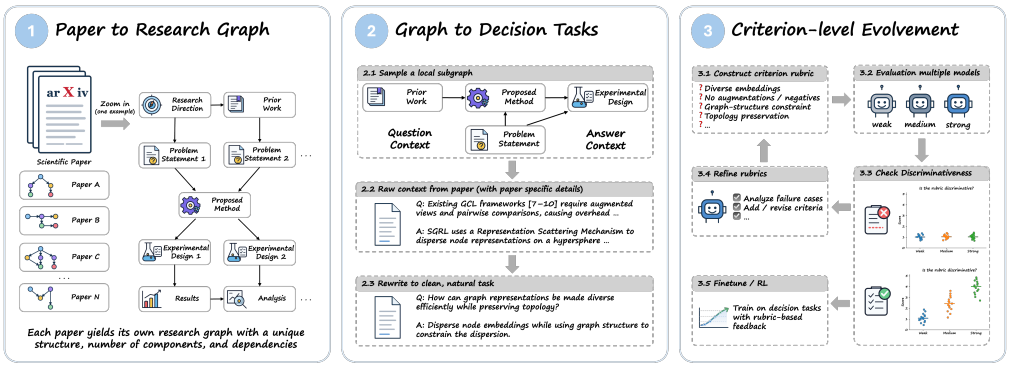

Figure 1: **Overview of Paper-Extracted Surrogate Task Optimization (PESTO).** (1) Papers are represented as research graphs. (2) Selected nodes supply question context and reference-answer evidence, rewritten into self-contained research-decision tasks. (3) Reference answers support rubrics, refined with multiple models' responses to better distinguish answer quality. Question-rubric pairs provide reinforcement learning rewards.

formulating a problem, proposing a method, designing experiments, and interpreting their results. We use these connections to construct shorter-horizon tasks in which part of a paper provides the current research context and the model must propose an appropriate next step. Narrowing each task makes the individual decision easier to evaluate and train, while constructing a diversity of tasks and supervisions across many papers exposes the model to a broad range of ideas and research approaches. We generate criterion-level rubrics for each task and refine them using responses from multiple models to remove trivial or brittle criteria. We then train on these tasks with RL, using rubrics as reward. We call this approach PESTO, or "Paper-Extracted Surrogate Task Optimization".

We train a 27B-parameter model with PESTO on tasks derived from ICML, ICLR, NeurIPS, and arXiv AI papers. The central question is whether training on these short-horizon surrogate decisions transfers to research settings that are substantially more open-ended than the training tasks themselves. We find that RL not only improves performance on held-out research questions, but also improves downstream outcomes when the model's proposals are implemented and evaluated. These gains extend from small, code-level research changes to broader project-level idea generation, where PESTO explores a wider range of approaches. Finally, we use PESTO to propose and iteratively refine an extension of a recent, unseen ICML 2026 oral paper, obtaining a distinct method that improves upon the reproduced results of the original paper. Our contributions are a framework for turning scientific papers into trainable research-decision problems and, more importantly, execution-grounded evidence that training on these short-horizon surrogate tasks can improve research guidance in substantially more open-ended settings, without using research execution as the training reward.

2 Related Work

AI for research. Recent systems use language models to automate research workflows, from literature retrieval and idea generation to experiments and manuscript writing (Lu et al., 2024; Yamada et al., 2025), or iteratively generate and test hypotheses around researcher-specified goals (Gottweis et al., 2026; Feng et al., 2026). Evaluation spans proposal assessments on reference papers or expert judgments (Qiu et al., 2025; Chen et al., 2026c;b) and execution-grounded benchmarks of paper replication, machine-learning systems improvements (Starace et al., 2025; Lyu et al., 2026; Si et al., 2026; Kirgis et al., 2026). Complementary work uses scientific literature to supervise document understanding and structured research relations (Wadden et al., 2025; Wu et al., 2026), idea and hypothesis generation (Li et al., 2024; O'Neill et al., 2025; Fan et al., 2026), and proposals and tool-using trajectories reconstruction (Wang et al., 2026a; Zhao et al., 2026). Most related to us, GIANTS trains models to anticipate a downstream paper's core insight from its parent papers (He-Yueya et al., 2026); ResearchBench and MOOSE-Star decompose discovery into inspiration retrieval and hypothesis composition tasks (Liu et al., 2026; Yang & Bing, 2026). Closest to our setting, Goel et al. (2025) extract research goals and grading criteria to train research-plan generation, while concurrent work PaperGym trains models to generate complete research plans from a paper's goals and background (Wang et al., 2026c). Our study addresses a complementary question: whether training on paper-derived surrogate decisions improves research outcomes when the resulting proposals are implemented, executed, and revised using experimental feedback. Building on paper-derived rubric supervision, we organize training around individual research decisions using paper dependency graphs. Not only does our model improve its rubric accuracy on a held-out set, but doing so also improves downstream open-ended research, when paired with execution in the loop.

Rewards and diversity in open-ended generation. Progress in open-ended research requires rewarding diverse ideas, hypotheses, and approaches, a desideratum shared with creative writing. Prior work learns rewards from preferences over creative outputs (Li et al., 2026b), quality preferences with response rarity or distinctiveness (Lanchantin et al., 2025b; Chung et al., 2025). However, applying these methods to scientific research is difficult: preferences over a proposal's presentation or plausibility may not reflect its scientific value, which is often revealed after substantial execution. Beyond overall preferences and distinctiveness, rewards can assess specific answer aspects through explicit criteria. For tasks without automatic verifiers, prompt-specific rubrics and checklists supply criterion-level rewards (Gunjal et al., 2026; Viswanathan et al., 2026), document-based pipelines synthesize question-answer-rubric triples (Bhattarai et al., 2026; Li et al., 2026a), and adaptive methods refine or evolve criteria during optimization (Zhang et al., 2026; Shen et al., 2026; Wang et al., 2026b). Related efforts assess research ideas without execution: EvoIdeator combines checklist-grounded rewards with actionable feedback (Sauter et al., 2026), while Tong et al. (2026) use citation-based community preferences to train a research-idea judge and generation policy. Optimizing these signals risks exploiting evaluator weaknesses to increase reward without improving scientific quality (Gao et al., 2023; Mahmoud et al., 2026), and concentrating on a narrow set of rewarded responses, reducing the diversity needed for open-ended exploration (Lanchantin et al., 2025b; Chung et al., 2025). Our approach addresses the diversity challenge through the construction of the training tasks. Using logical dependencies between paper statements, we construct focused questions about individual research decisions. Applying this construction across papers brings different approaches to related decisions into a shared task structure. This design aims to retain the breadth of scientific literature while providing focused questions that can be optimized with rubric rewards.

3 PESTO: From Papers to Trainable Research Problems

Our goal is to train models to give better research directions: given the current findings in a research project, what should be tried next? In principle, we could train models by letting them propose ideas, execute and refine them, and learn from their eventual consequences. In practice, experiments are slow and expensive, particularly in AI. Outcomes also provide weak feedback on the decisions that led to them: a negative result may still be useful if it rules out an explanation or suggests a next step. Scientific papers offer another source of training data. They contain many of the ***research decisions*** we want models to learn: identifying gaps, proposing methods, designing experiments, and interpreting results. Training to predict paper text, however, does not directly train these decisions, as we also show empirically. Papers present finished research rather than each decision along the way, and typically describe only the approach ultimately pursued. We therefore turn papers into explicit research questions. Each task presents the information available at some point in a paper and asks the model to make a research decision, while later parts of the paper provide evidence for evaluating it. Across many papers, this exposes the model to different approaches to related research problems.

As shown in Figure 1, we break each paper into components such as prior work, limitations, methods, etc., and record their dependencies. We use these components to construct self-contained research questions and answers, then turn each answer into evaluation criteria. We refine these criteria using responses from several models to remove vague or brittle requirements. Finally, we train with RL using the ratio of satisfied criteria as reward, akin to standard rubric-based RL. This provides scalable feedback on individual research decisions without executing a full research project for each training example. We next describe the paper corpus, task and rubric construction, and training procedure.

3.1 Training And Evaluation Corpora

We combine two complementary training sources, both restricted to papers published in 2024-2025. First, we collect 1,625 oral or spotlight papers from ICLR, ICML, and NeurIPS conferences as an initial source of research-decision tasks. We then supplement this set with arXiv papers to increase the number and variety of research examples. The arXiv candidate pool spans nine categories (`cs.LG`, `cs.AI`, `cs.CL`, `cs.CV`, `cs.NE`, `cs.RO`, `cs.IR`, `cs.MA`, and `stat.ML`). After excluding surveys, duplicates of the curated conference set, and near-duplicate titles, approximately 269,000 candidates remain. To select a manageable training corpus, we rank these papers using a weighted combination of citation impact, venue, author citation impact, content similarity, and recency:

$$s(p) = .40s_{\text{cohort}} + .20s_{\text{venue}} + .15s_{\text{author}} + .20s_{\text{semantic}} + .05s_{\text{recent}}. \quad (1)$$

Here, s_{cohort} is a paper's citation percentile within its publication-month cohort, reducing the advantage of older papers. Motivated by citation-normalization work (Radicchi et al., 2008), we normalize by publication time within the broader machine-learning pool, but not by field.

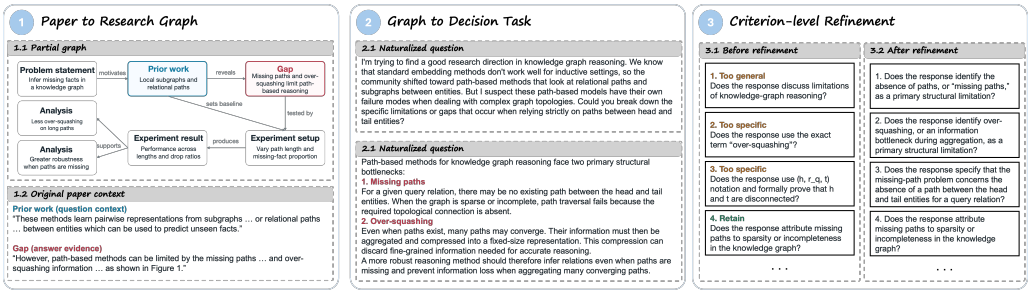

Figure 2: **A research-decision task constructed from KnowFormer (Liu et al., 2024).** (1) The partial research graph links prior work on path-based knowledge-graph reasoning to the gap it reveals. The prior-work node provides question context, and the gap node provides a reference answer. (2) The selected passages are rewritten into a self-contained question about structural limitations and a reference answer identifying missing paths and over-squashing. (3) Rubric refinement replaces an overly general criterion or brittle terminology with checks of the underlying technical content, while retaining a valid criterion about graph sparsity and incompleteness.

The remaining terms capture venue tier, author impact via h-indices (Hirsch, 2005), similarity to the curated conference-paper centroid in scientific-document embedding space (Cohan et al., 2020), and recency. The weights are heuristic: we place the greatest weight on cohort-normalized citations, equal secondary weight on venue and content similarity, and a smaller weight on recency. This score is neither intended as a measure of research quality nor a uniquely correct ranking method, but a practical heuristic for constructing a training corpus with broad coverage. We retain the highest-ranked arXiv papers and report their domain coverage in Figure 3.

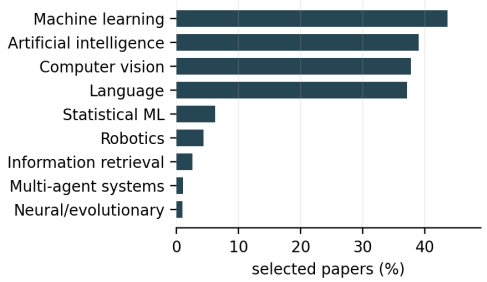

Figure 3: **Domain coverage of the selected 10K-paper corpus.** The corpus includes papers from all nine arXiv categories. Papers can have multiple category labels, so percentages need not sum to 100%.

Held-out test set. We use 221 ICLR 2026 papers with *Oral* ratings as our held-out evaluation set. We cross-check these papers against our training corpus and remove matching records.

3.2 Turning Papers Into Question-Rubric Tuples

From papers to dependency graphs. To construct research tasks, we identify each paper's components and dependencies. We convert each paper PDF into Markdown using *gemini-3-flash-preview* for affordable multimodal parsing at corpus scale while preserving equations, tables, and figure descriptions. We use *gemini-3.1-pro-preview* for subsequent stages, including graph construction, question-answer generation, rubric generation and refinement. We then split the Markdown file into sections and represent each section as a node in a directed acyclic graph. Each node contains the corresponding passage from the paper. Drawing on previous work on identifying the functions of passages in scientific papers (Liakata, 2010; Yang & Li, 2018), we assign a label to each node describing its *role* in the paper, such as motivation, method idea, method detail, experimental result or limitation. The edges in this dependency graph record their logical dependencies. Figure 4 shows the distribution of these labels.

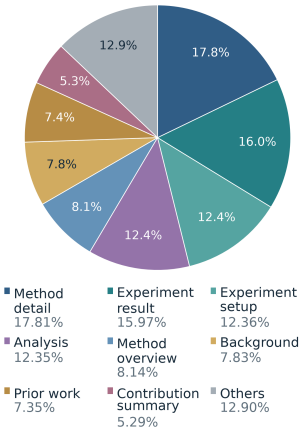

Figure 4: **Semantic roles in paper graphs.** Node distribution across the broad corpus. Roles below 5% each form *Others*: gap, motivation, problem statement, limitation, key insight, future direction, and hypothesis.

Figure 2 provides an example of this construction using *KnowFormer* (Liu et al., 2024). The partial graph in panel 1.1 connects prior work on path-based knowledge-graph reasoning to its limitations, experimental setup, results, and analysis. Panel 1.2 shows the paper passages associated with the prior-work and gap nodes, which supply the context and answer evidence for the example task.

From graph nodes to question-answer pairs. We select graph nodes as question context and reference-answer evidence. Predefined rules map node roles to research decisions, such as identifying gaps from prior work or designing experiments from a method. In the *KnowFormer* example (Figure 2,

Table 1: **Baseline success rates by research decision.** Mean rubric satisfaction is higher for literature gaps than for experiment design or idea development across all three models.

Model	Literature → gap	Problem/gap → method	Method → experiments	Direction → idea
Claude Opus 4.8	**61.9%**	**54.6%**	**48.2%**	**50.1%**
Gemini 3.1 Pro Preview	45.2%	47.7%	36.9%	36.4%
OpenAI o4-mini	41.0%	40.2%	29.8%	29.0%

panels 1-2), the *literature_to_gap* rule selects the blue prior-work node as question context and its direct child, the red gap node, as answer evidence. The question asks which structural limitations remain in path-based knowledge-graph reasoning, and the reference answer identifies missing paths and over-squashing. We also construct mixed-context pairs using four cross-stage selection rules, extending coverage beyond standard research operations. Appendix B gives the deterministic selection procedure. Figure 5 shows the distribution of pair types among the final training questions.

Question generation and rubric refinement. An LLM rewrites each question and reference answer, preserving technical content while removing paper self-reference and citation-dependent phrasing (Figure 2, panel 2.2). We then generate ten independently judgeable binary criteria from the reference answer, following prior work on reference-based rubrics for RL (Shao et al., 2025). LLM-generated grading schemes and model-based judges have also been used for rubric-guided learning in mathematical proof generation (Qu et al., 2026). We similarly use rubrics as tractable surrogate feedback, rather than an exhaustive characterization of scientifically valid answers. Our execution-based evaluations test whether learning from this feedback transfers beyond rubric scores.

A major challenge is determining whether the generated rubrics are suitable for effective RL without running RL for every design choice. We therefore use responses from Claude Opus 4.8, GPT o4-mini, and Gemini 3.1 Pro Preview on the same tasks. We score these responses with the generated rubrics and refine the criteria based on their pass patterns: we retain criteria that distinguish responses, make universally passed criteria more specific using the reference answer, and repair brittle requirements in criteria that all models fail. Figure 2, panel 3, shows several examples. A general check for discussing limitations becomes a specific check for identifying missing paths; an exact-term requirement for "over-squashing" is broadened to accept an explanation of the underlying information bottleneck; and a notation-and-proof requirement becomes a check for explaining why no path exists between the query entities. A criterion linking missing paths to graph sparsity or incompleteness is retained unchanged. We measure the effect of these refinements by re-scoring the same frozen responses. Each response receives the fraction of rubric criteria it satisfies: $R(x, y, r) = \frac{1}{|r|} \sum_{c \in r} \mathbf{1}[J(x, y, c) = \text{YES}]$, where x is the question, y the response, r the rubric, and J the judge.

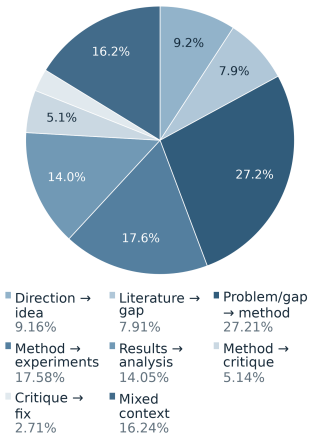

Figure 5: **Problem type distribution.** The training corpus is well spread across question types.

Final dataset. After filtering invalid records, our training corpus contains 336,840 question-rubric records from 10,596 papers, drawn from 2024-2025 sources in Section 3.1. Related questions are associated with an average of 2.18 reference answers as detailed in Appendix B.1, providing corpus-level answer multiplicity while retaining a separate rubric for each record. Our held-out ICLR 2026 evaluation set contains 221 papers and 3,964 question variants with rubrics. Upon publication, we will release both datasets, including paper graphs, question-answer pairs, and refined rubrics.

Table 1 breaks down success rates for four types of research decisions. All three proprietary models shown score higher on identifying literature gaps and developing methods than on designing experiments or developing ideas from research directions, indicating varying difficulty across capabilities. Table 2 reports baseline performance, measured by *success rate*: the mean fraction of rubric criteria satisfied. The tasks remain challenging: across 3,964 scored evaluation problems, Claude Opus 4.8 satisfies every criterion on only 12.5% and none on 8.3%. The Qwen3.6-27B base model satisfies every criterion on 6.9% and none on 15.3%, leaving substantial room for improvement.

3.3 Supervised fine-tuning and rubric-guided reinforcement learning

Following prior work combining supervised fine-tuning with rubric-guided RL (Qu et al., 2026), we initialize the policy π_θ from Qwen3.6-27B (Qwen Team, 2026) and train in two stages. We first

perform SFT on questions and reference answers rewritten by the Qwen base model from one half of the training papers, then apply rubric-guided RL to the other half. The two halves are disjoint, with all papers in the curated high-quality oral subset from NeurIPS, ICML, and ICLR assigned to the RL portion. For RL, we use an asynchronous implementation of CISPO (Chen et al., 2025) with an FP32 LM head, sampling eight responses for each of 32 questions per batch. Each response receives the fraction of audited rubric criteria it satisfies as its reward. We normalize rewards within each group and apply clipped importance weights to account for differences between the rollout and trainer policies. We use gpt-oss-20B as the judge for online rewards and Gemini 3.1 Pro Preview for held-out evaluation. During RL, reference answers influence training only through the generated rubrics. Appendix A gives the objective and the full training and judging configuration.

4 Experimental Evaluation

The goal of our experiments is to test whether training on research tasks derived from papers improves a model's ability to guide research by proposing ideas, designing experiments, and analyzing results. We organize our evaluations along a progression of increasing scope of research decisions delegated to the model: from answering specific research questions similar to those encountered during training, to making research decisions within a narrowly prescribed implementation scope, all the way to developing research extensions of a paper, end to end. As this scope increases, our evaluation correspondingly shifts from directly assessing the quality of the model's outputs to measuring the downstream outcomes of the model's decisions. Concretely, we consider four evaluation settings.

(1) *Question answering **(QA)**:* The model answers a specific research question given part of a paper's context, such as identifying a gap, proposing a method or experiment, and analyzing results. These questions are analogous in form to those in our training set, but are constructed from held-out papers. We directly evaluate the quality of the model's response using criterion-level rubrics.

(2) *Local guidance **(LG)**:* The model is given a high-level research objective, typically expressed as reducing a loss or improving efficiency, together with a narrowly specified implementation scope (e.g., modifying only the neural-network layers defined in a single file). Because the objective is directly measurable, we evaluate the model through the score achieved after implementing its proposal, normalized relative to the best theoretically achievable score. We study this setting on several tasks from the MLS-Bench (Lyu et al., 2026) and NatureBench (Wang et al., 2026d) benchmarks.

(3) *Research toward a fixed objective **(FO)**:* This setting expands upon **(2)** by retaining a measurable objective while substantially broadening the space of approaches available to the model. Rather than making decisions within a narrowly prescribed implementation scope, the model can choose which research directions to pursue and which components of the system to modify in order to improve the target metric. We study two tasks: language-model pretraining with nanoGPT and math post-training of small LLMs with GRPO (Si et al., 2026). We evaluate both the effectiveness of the resulting approaches on the target objective and the diversity of research approaches the model discovers.

(4) *Open-ended paper extension **(OE)**:* Finally, we ask the model to attempt the "next paper" building on existing work. Starting from a full paper and its codebase, the model is given freedom to identify a limitation worth addressing, formulate a proposed extension, design experiments, and use the resulting evidence to refine its proposal. This setting connects gap identification, method design, experimental planning, and evidence-driven revision within the supplied paper's codebase and evaluation framework. We study this setting using AI Engram (Kwon et al., 2026), an ICML 2026 oral paper that was excluded from our training set, with a fixed executor that selects among the model's proposals and carries out the corresponding implementation and experiments. We selected this paper in part because its experiments can be executed substantially more quickly (i.e., a few hours) than those of many other ICML 2026 oral papers.

Experimental setup. We train `PESTO` from Qwen3.6-27B using supervised fine-tuning followed by reinforcement learning (SFT + RL). We split the training corpus into two disjoint halves of approximately 5,000 papers each, with all curated NeurIPS, ICML, and ICLR oral papers before 2026 in the RL split. We first perform SFT on question-reference-answer pairs from one half, rewritten by Qwen3.6-27B to avoid a shift in the response style that is known to hamper SFT, and then apply rubric-guided RL to the other half. We compare against the base Qwen3.6-27B model, continued pretraining (CPT) on the full text of all training papers, two SFT baselines using pairs rewritten by Gemini 3 Flash or Qwen3.6-27B, and a "high-compute" RL-only ablation trained with refined rubrics on *all* papers. Each baseline uses the full corpus; `PESTO` covers it across its two stages. SFT (Gemini)

Table 2: **Performance of PESTO under the (QA) setting.** Success rates using non-thinking mode where available and native mode otherwise, including differences from base Qwen3.6-27B. Green/pink indicate gains/losses; darker shading marks larger changes (< 4, 4–< 8, ≥ 8 points). Observe that PESTO attains the highest observed average rubric satisfaction, with RL-only training performing similarly; the evaluated SFT-only and CPT baselines do not improve the average over the base model.

Model	Dir.→Idea	Lit.→Gap	Gap→Meth.	Meth.→Exp.	Res.→Anal.	Mixed	Avg. success
API models							
GPT-5.5	41.12	58.87	**58.05**	40.17	52.54	51.06	52.65
Gemini 3.5 Flash	39.88	53.10	51.45	43.47	50.42	49.25	49.15
Gemini 3.1 Pro	36.41	45.20	47.69	36.91	46.97	42.51	44.45
Claude Opus 4.8	**50.07**	**61.89**	54.62	48.22	48.82	52.88	52.71
OSS + trained models							
Qwen3.6-27B base	29.32	45.84	43.32	33.60	43.65	43.09	41.02
Qwen3.6-35B-A3B	25.12	40.46	39.53	31.62	42.82	37.94	37.65
SFT (Gemini)	14.13 (−15.19)	17.75 (−28.09)	20.42 (−22.90)	19.17 (−14.43)	20.32 (−23.33)	17.64 (−25.45)	19.40 (−21.62)
SFT (Qwen)	25.24 (−4.08)	34.50 (−11.34)	36.59 (−6.73)	28.78 (−4.82)	36.44 (−7.21)	33.46 (−9.63)	34.16 (−6.86)
CPT	29.86 (+0.54)	42.97 (−2.87)	42.51 (−0.81)	35.68 (+2.08)	42.18 (−1.47)	39.71 (−3.38)	40.30 (−0.72)
RL	43.02 (+13.70)	61.52 (+15.68)	56.45 (+13.13)	46.50 (+12.90)	54.11 (+10.46)	53.83 (+10.74)	53.66 (+12.64)
PESTO ***(Ours)***	42.14 (+12.82)	60.34 (+14.50)	56.15 (+12.83)	**48.64** (+15.04)	**55.32** (+11.67)	**54.73** (+11.64)	**54.02** (+13.00)

and SFT (Qwen) identify the model that rewrites training pairs; both fine-tune Qwen3.6-27B. We also report frontier API-model results where available. For the execution-based evaluations in Results 2-3, an idea model proposes a change in natural language and a separate Codex implementer translates it into code, separating research guidance from direct code-generation ability. Task-specific evaluation protocols are described below, with the nanoGPT and GRPO workflow in Appendix E.

Result 1 [(QA)]: RL improves research decisions across tasks. We first evaluate PESTO on direct (QA) tasks derived from a held-out subset of ICLR 2026 oral papers. Table 2 compares our SFT + RL recipe against SFT and CPT, all starting from the same Qwen3.6-27B base model. SFT followed by rubric-guided RL improves performance across all six task categories, increasing average success from 41.02% to 54.02%. For example, Literature → Gap improves from 45.84% to 60.34%, while Method → Experiments improves from 33.60% to 48.64%. In contrast, SFT on either Gemini- or Qwen-generated data degrades performance across all categories, while CPT on the paper text remains close to the base model on average (40.30% vs. 41.02%) despite training on 230M tokens. RL alone reaches 53.66% average rubric satisfaction, close to PESTO's 54.02%. We adopt SFT followed by RL to incorporate part of the corpus without incurring rollout-and-judging costs for that stage. Under the evaluated configurations, both recipes improve substantially over the base model, whereas the reported SFT-only and CPT baselines do not.

Result 2 [(LG)]: PESTO improves on MLS-Bench (Lyu et al., 2026) and NatureBench (Wang et al., 2026d). Next, we evaluate PESTO on (LG) tasks. We randomly select a subset of ML-oriented tasks from MLS-Bench and NatureBench. MLS-Bench specifies a research objective and restricted code-editing scope, while NatureBench provides scientific coding tasks with implementation guidance. For both, our trained model proposes a change in natural language, Codex implements it, and the benchmark evaluator scores the resulting program. Figure 6 reports the mean and maximum scores across eight parallel proposals. MLS-Bench reports normalized task scores, while for NatureBench we report performance ratios relative to the paper-reported reference, computed as one plus the benchmark's original improvement score. A NatureBench score of 1.0 therefore matches the SOTA performance, while values above 1.0 improve upon it. Higher is better in both benchmarks, although score magnitudes should be compared within tasks rather than across benchmarks.

On MLS-Bench (Figure 6, left), PESTO improves mean performance over Qwen3.6-27B on three of four task families and maximum performance on all four. Pooling mean score increases from 0.1635 to 0.2194, and forecasting mean score from 0.2385 to 0.2526. Dimensionality reduction presents a trade-off: the mean decreases from 0.1555 to 0.1433, while the maximum increases from 0.3369 to 0.3474. Thus, the gains extend across most task families but are not uniform across proposals.

On NatureBench (Figure 6, right), PESTO improves both mean and maximum performance ratios over the base idea model on all three tasks. The largest mean gain is in spatial point-cloud clustering, where the ratio increases from 0.9532 to 1.0730 and the maximum increases from 1.1141 to 1.1733. Thus, PESTO exceeds the paper-reported clustering reference both on average and at its best. Mean ratios also improve for MOF property prediction, from 0.9128 to 0.9397, and cross-modal cell matching, from 0.8762 to 0.8923. However, even the maximum ratios on these two tasks remain below 1.0 (0.9611 and 0.9251, respectively), so PESTO narrows the gap to their paper-reported

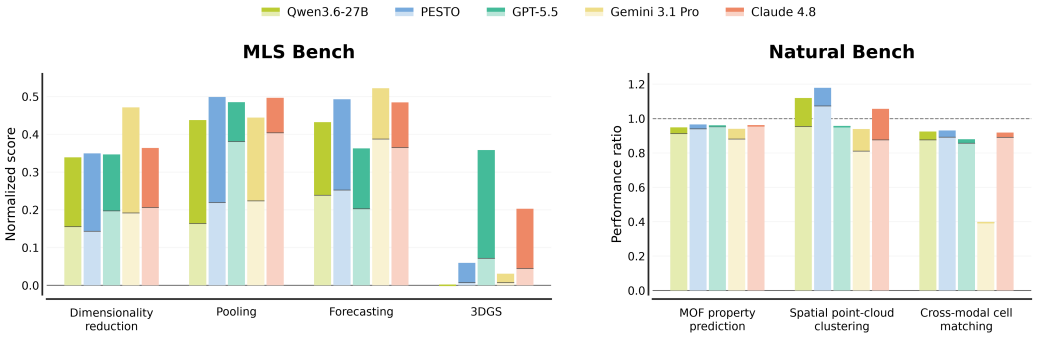

Figure 6: **Performance of PESTO on MLS-Bench and NatureBench.** Each task shows one bar per model, with consistent colors across panels. Light segments show mean scores; dark segments extend to the maximum. Higher is better. NatureBench ratios equal one plus the original improvement score; the dashed line at 1.0 marks the paper-reported reference. API-model scores are shown for reference.

references without surpassing them. Across MLS-Bench and NatureBench, PESTO improves the mean score on six of seven task families and the maximum score on all seven. These results support the central premise of this work: training on research tasks derived from a broad collection of papers can improve a model's ability to make research decisions that translate into downstream gains.

Result 3 [(FO)]: PESTO improves proposal outcomes, with qualitative evidence of broader exploration. We evaluate both research environments released by Si et al. (2026), nanoGPT pretraining and GRPO post-training, under our fixed-budget protocol. In nanoGPT, the objective is to improve language-model pretraining on FineWeb (Penedo et al., 2024), measured by held-out validation cross-entropy. In GRPO, the objective is to improve reasoning post-training of Qwen2.5-Math-1.5B (Yang et al., 2024), measured by strict held-out correctness on 500 MATH (Lightman et al., 2024) problems. These tasks retain fixed code and evaluation constraints, but leave the choice of research mechanism to the idea model, including changes to architecture, optimization, curricula, or the training objective.

Table 3: **Results on (FO) tasks.** Mean and best scores across 8 parallel proposals per task; best denotes lowest nanoGPT loss or highest GRPO reward (held-out accuracy).

	nanoGPT loss ↓		GRPO reward ↑	
Model	Mean	Best	Mean	Best
Qwen3.6-27B	3.4431	3.2532	0.3823	0.4760
PESTO	**3.2691**	**3.2373**	**0.4580**	**0.4840**

For each task and idea model, we evaluate eight proposals generated from the same initial code context without feedback from earlier experiments. A fixed Codex implementer applies each accepted proposal to a fresh environment under a shared task-specific protocol. nanoGPT uses a three-hour training limit; GRPO uses a three-hour execution timeout. We report the mean and best score across the eight proposals. Each GRPO proposal is scored using its last completed checkpoint, rather than the best. Appendix E gives the full protocol, prompts, per-proposal results, and available excerpts of the highest-scoring responses. PESTO improves both the mean and best score on both tasks (Table 3). On nanoGPT, mean validation loss decreases from 3.4431 to 3.2691, and the best loss decreases from 3.2532 to 3.2373. On GRPO, mean held-out reward increases from 0.3823 to 0.4580, while the best proposal increases accuracy from 47.6% to 48.4%. *The mean gains are substantially larger than the best-score gains, reflecting in part the base model's poorly performing candidates.*

What do the proposed research directions look like? On nanoGPT, PESTO's best direction replaces absolute positional embeddings with rotary position embeddings. On GRPO, its best direction normalizes rewards by response length before computing group-relative advantages, while leaving the held-out grader unchanged. Across eight proposals per model and task, PESTO shows qualitative evidence of broader mechanism coverage (Appendix E, Table 6–7). Six base-model nanoGPT proposals focus on gradient scaling or layer-wise learning rates, whereas PESTO also explores context curricula, attention regularization, and positional representations. Six base-model GRPO proposals modify length-based reward or advantage scaling, whereas PESTO also explores self-verification, entropy regularization, and position-dependent credit assignment. Note that these sampled proposals do not establish a population-level diversity gain or novelty relative to the literature.

Result 4 [(OE)]: Extending an ICML oral paper. We test whether a model can connect research decisions *end-to-end* (Figure 7): identifying a gap, proposing a method, designing experiments, interpreting results, and revising the method. We study AI Engram (Kwon et al., 2026), an ICML 2026 oral paper, comparing Qwen3.6-27B with PESTO. Both models start from the same paper, code context, and reproduced adaptive Engram baseline. At each research-decision stage, the advising model generates 16 candidate next steps. A fixed Codex selector chooses among them and executes the selected methods and experiments, returning feedback to the advising model. We repeat this complete end-to-end workflow eight times per model under the same selection-and-execution protocol and report each model's best final trajectory, selected according to the highest final Overall score. Interestingly, in their respective best-of-eight trajectories, both models independently identify the same gap: relative projection magnitude does not directly capture whether a layer is selectively associated with forgetting rather than retention. Their trajectories diverge in how they translate this observation into a method and how they revise that method after observing experimental results for the next round.

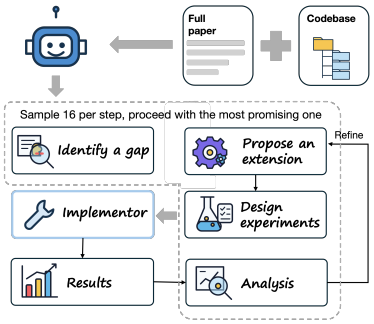

Figure 7: **Open-ended paper extension (OE).** The model iteratively generates guidance, with selected ideas executed to inform subsequent rounds.

> **Best-of-eight research trajectories (Codex analysis).** Both selected trajectories identify the same gap and construct a selectivity score from the ratio of forget- to retain-conditioned projection norms. The base model proposes an inverse gate, $1/(1+0.5s_\ell)$, assigning *smaller* corrections to more forget-specific layers. Although its subsequent reasoning notices this inconsistency, it retains the selected equation. In contrast, PESTO proposes a correction that increases with selectivity, matching the motivation behind the score. Initial experiments then reveal a problem: the base model under-edits, while PESTO over-edits and violates the exact-match (EM) constraint. Both models respond by normalizing their scaling quantities by their layerwise means; PESTO additionally clips the resulting correction factors. Thus, the models differ not only in their initial proposal, but also in how they use experimental feedback to refine it. Appendix F provides the complete results.

The extension improves upon the reproduced baseline. As shown in Table 4, the best-of-eight PESTO extension increases Overall from 0.8137 to 0.8396 and Privacy from 0.6404 to 0.6915, while reducing EM from 0.0223 to 0.0191. The paired row-bootstrap 95% confidence interval for the Privacy gain is $[+0.0401, +0.0621]$, conditional on the selected trajectory (Appendix F). Memorization and Utility decrease by less than 0.0011 each and remain within the prespecified preservation thresholds. The base model recovers from its initial failure through revision but remains below the reproduced baseline on Overall and Privacy. This case study illustrates that PESTO can connect problem identification, method design, experimentation, and evidence-driven revision into a research trajectory that yields a measurable improvement over a recent published baseline.

Table 4: **Extending AI Engram (OE).** Best-of-eight trajectory results for each advising model, compared with the reproduced adaptive Engram baseline. All metrics in each model row come from the same selected trajectory. Higher is better except for EM.

Method	Overall	Privacy	Mem.	Utility	EM↓
Adaptive Engram	0.8137	0.6404	**0.9593**	0.9235	0.0223
Qwen3.6-27B	0.8014	0.6176	0.9569	**0.9265**	0.0388
PESTO	**0.8396**	**0.6915**	0.9590	0.9225	**0.0191**

5 Discussion and Perspectives on Future Work

We introduce Paper-Extracted Surrogate Task Optimization (PESTO), which converts scientific papers into research-decision tasks and trains LLMs with rubric rewards, avoiding costly execution-grounded training. Across four evaluation settings, PESTO improves models' ability to propose useful research steps, with qualitative evidence of broader exploration. Separating idea generation from implementation shows that open models can compete with proprietary models in several evaluated settings. These gains do not establish independent progress beyond human research: NatureBench results do not consistently exceed human-developed references, and our illustrative paper extension builds on a human-designed method. Our findings support paper-derived training as a way to help researchers explore and refine ideas. Future work could incorporate execution feedback and researcher judgments, jointly train idea-generation and execution models, extend beyond machine learning, and reward scientifically valid alternatives to published solutions. Evaluations with human researchers should test whether these capabilities reduce experimentation costs and improve research outcomes.

Ethics Statement

This work studies the use of language models to support AI research by generating and evaluating research decisions. Our experiments do not involve human subjects or private personal data. The training data are derived from publicly available scientific papers, and the resulting models may inherit biases, errors, or other limitations present in these sources and in the language models used for data construction and evaluation. More capable research-assistance systems may also accelerate the development of both beneficial and potentially harmful technologies. We therefore view the methods studied here as tools for assisting researchers rather than replacing human judgment and oversight. We plan to release research artifacts with appropriate documentation and safeguards, and encourage their responsible use in accordance with applicable licenses and research-ethics standards.

Reproducibility Statement

We provide detailed descriptions of the data construction and training procedure in Section 3 and Appendix A-B, including the research-task construction algorithm, RL objective, hyperparameters, and judging protocol. Appendices C–D provide worked examples of the generated questions, reference answers, and rubrics. Section 4 and Appendices E–F describe the evaluation protocols, prompts, implementation workflow, per-proposal results, and the open-ended paper-extension experiment. Upon publication, we plan to release the constructed datasets, including paper graphs, question–answer pairs, and refined rubrics, as stated in Section 3.2.

LLM Usage Disclosure

The research idea underlying this work was developed by the authors without LLM assistance. We used large language models (LLMs) to help implement the research workflow and generate synthetic datasets, including the question-answer pairs and rubrics described in Section 3.2. The LLM-generated research proposals and decisions reported in our experiments are outputs of the systems under evaluation and are themselves subjects of this study. We also used LLMs to polish author-written text, edit figures, and retrieve and identify relevant literature. We did not use LLMs to draft sections of the paper or prove mathematical claims. The authors take responsibility for the final content of this work, including its text, figures, references, datasets, code, claims, and other artifacts produced with the assistance of generative AI.

References

Manish Bhattarai, Ismael Boureima, Nishath Rajiv Ranasinghe, Scott Pakin, and Dan O'Malley. Rubric-grounded rl: Structured judge rewards for generalizable reasoning. *arXiv preprint arXiv:2605.08061*, 2026.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Advances in Neural Information Processing Systems*, 38:57654–57689, 2026a.

Zhiyu Chen, Keyu Zhao, Jigao Fu, Dong Liang, Yanbiao Wu, Jiaoyang Li, Haidong Xue, Xinhua Zeng, Yuanyi Zhen, Fengli Xu, et al. Ideation arena: Evaluating llm generated research ideas with battle-style human expert assessment. *arXiv preprint arXiv:2608.29696*, 2026b.

Ziyu Chen, Yilun Zhao, and Arman Cohan. Measuring the gap between human and llm research ideas. *arXiv preprint arXiv:2607.01233*, 2026c.

John Joon Young Chung, Vishakh Padmakumar, Melissa Roemmele, Yuqian Sun, and Max Kreminski. Modifying large language model post-training for diverse creative writing, 2025. URL https://arxiv.org/abs/2503.17126.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 2270–2282, 2020.

Tianyu Fan, Fengji Zhang, Yuxiang Zheng, Bei Chen, Xinyao Niu, Chengen Huang, Junyang Lin, and Chao Huang. Deepinnovator: Triggering the innovative capabilities of llms. *arXiv preprint arXiv:2602.18920*, 2026.

Shiyang Feng, Runmin Ma, Xiangchao Yan, Yue Fan, Yusong Hu, Songtao Huang, Shuaiyu Zhang, Zongsheng Cao, Tianshuo Peng, Jiakang Yuan, et al. Internagent-1.5: A unified agentic framework for long-horizon autonomous scientific discovery. *arXiv preprint arXiv:2602.08990*, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pp. 10835–10866. PMLR, 2023.

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, 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.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations*, volume 2026, pp. 127924–127945, 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.

Jorge E Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences*, 102(46):16569–16572, 2005.

Jiewen Hu, Thomas Zhu, and Sean Welleck. minictx: Neural theorem proving with (long-)contexts, 2025. URL https://arxiv.org/abs/2408.03350.

Peter Kirgis, Sayash Kapoor, Andrew Schwartz, Stephan Rabanser, David Africa, Konstantinos Voudouris, Viet Nguyen, Toby Pilditch, Magda Dubois, Harry Coppock, et al. Can ai agents conduct open-ended ai research? early evidence from two case studies. *arXiv preprint arXiv:2607.27191*, 2026.

Jea Kwon, Dong-Kyum Kim, Jiwon Kim, Yonghyun Kim, Woong Kook, and Meeyoung Cha. Ai engram: In search of memory traces in artificial intelligence. *arXiv preprint arXiv:2606.14997*, 2026.

Jack Lanchantin, Angelica Chen, Shehzaad Dhuliawala, Ping Yu, Jason Weston, Sainbayar Sukhbaatar, and Ilia Kulikov. Diverse preference optimization. *arXiv preprint arXiv:2501.18101*, 2025a.

Jack Lanchantin, Angelica Chen, Shehzaad Dhuliawala, Ping Yu, Jason Weston, Sainbayar Sukhbaatar, and Ilia Kulikov. Diverse preference optimization, 2025b. URL https://arxiv.org/abs/2501.18101.

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

Xiaoyuan Li, Keqin Bao, Moxin Li, Yubo Ma, Yichang Zhang, Wenjie Wang, Fuli Feng, and Dayiheng Liu. Ares: Automated rubric synthesis for scalable llm reinforcement learning. *arXiv preprint arXiv:2605.23454*, 2026a.

Zhaoyan Li, Hang Lei, Yujia Wang, Lanbo Liu, Hao Liu, and Liang Yu. Rewarding creativity: A human-aligned generative reward model for reinforcement learning in storytelling, 2026b. URL https://arxiv.org/abs/2601.07149.

Maria Liakata. Zones of conceptualisation in scientific papers: a window to negative and speculative statements. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pp. 1–4, 2010.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264.

Junnan Liu, Qianren Mao, Weifeng Jiang, and Jianxin Li. Knowformer: revisiting transformers for knowledge graph reasoning. *arXiv preprint arXiv:2409.12865*, 2024.

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. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13187–13207, 2026.

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.

Bohan Lyu, Yucheng Yang, Siqiao Huang, Jiaru Zhang, Qixin Xu, Xinghan Li, Xinyang Han, Yicheng Zhang, Huaqing Zhang, Runhan Huang, et al. Mls-bench: A holistic and rigorous assessment of ai systems on building better ai. *arXiv preprint arXiv:2605.08678*, 2026.

Anas Mahmoud, MohammadHossein Rezaei, Zihao Wang, Anisha Gunjal, Bing Liu, and Yunzhong He. Reward hacking in rubric-based reinforcement learning. *arXiv preprint arXiv:2605.12474*, 2026.

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.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

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.

Yuxiao Qu, Amrith Setlur, Jasper Dekoninck, Edward Beeching, Jia Li, Ian Wu, Lewis Tunstall, Aviral Kumar, et al. Qed-nano: Teaching a tiny model to prove hard theorems. *arXiv preprint arXiv:2604.04898*, 2026.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

Filippo Radicchi, Santo Fortunato, and Claudio Castellano. Universality of citation distributions: Toward an objective measure of scientific impact. *Proceedings of the National Academy of Sciences*, 105(45):17268–17272, 2008.

Andreas Sauter, Yuyue Zhao, Jacopo Urbani, Wenxiang Hu, Zaiqiao Meng, Lun Zhou, Xiaohui Yan, and Yougang Lyu. Evoideator: Evolving scientific ideas through checklist-grounded reinforcement learning. *arXiv preprint arXiv:2603.21728*, 2026.

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.

William F Shen, Xinchi Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. Rethinking rubric generation for improving llm judge and reward modeling for open-ended tasks. *arXiv preprint arXiv:2602.05125*, 2026.

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

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.

Jingqi Tong, Mingzhe Li, Hangcheng Li, Yongzhuo Yang, Yurong Mou, Weijie Ma, Zhiheng Xi, Hongji Chen, Xiaoran Liu, Qinyuan Cheng, et al. Ai can learn scientific taste. *arXiv preprint arXiv:2603.14473*, 2026.

Vijay Viswanathan, Yanchao Sun, Xiang Kong, Meng Cao, Graham Neubig, and Sherry Wu. Checklists are better than reward models for aligning language models. *Advances in Neural Information Processing Systems*, 38:114728–114754, 2026.

David Wadden, Kejian Shi, Jacob Morrison, Alan Li, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, et al. Sciriff: A resource to enhance language model instruction-following over scientific literature. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6083–6120, 2025.

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*, 2026a.

Jianze Wang, Kunwang Zheng, Ying Liu, Yu Cao, Qilong Zhang, Jinlong Chen, Hua Yang, and Qianglong Chen. Serpo: Self-evolving rubric policy optimization for open-ended test-time reinforcement learning, 2026b. URL https://arxiv.org/abs/2607.26873.

Yuhan Wang, Zhengxi Lu, Yuchen Yan, Kaitao Song, Wenqi Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Papergym: Rubric-centered evolution for research-plan generation. *arXiv preprint arXiv:2608.31119*, 2026c.

Yuru Wang, Lejun Cheng, Yuxin Zuo, Sihang Zeng, Bingxiang He, Che Jiang, Junlin Yang, Yuchong Wang, Kaikai Zhao, Weifeng Huang, et al. Naturebench: Can coding agents match the published sota of nature-family papers? *arXiv preprint arXiv:2606.24530*, 2026d.

Yujun Wu, Dongxu Zhang, Xinchen Li, Jinhang Xu, Yiling Duan, Yumou Liu, Jiabao Pan, Qiyuan Zhu, Xuanhe Zhou, Jingxuan Wei, et al. Intern-atlas: A methodological evolution graph as research infrastructure for ai scientists. *arXiv preprint arXiv:2604.28158*, 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 and Sujian Li. Scidtb: Discourse dependency treebank for scientific abstracts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 444–449, 2018.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Zonglin Yang and Lidong Bing. Moose-star: Unlocking tractable training for scientific discovery by breaking the complexity barrier. *arXiv preprint arXiv:2603.03756*, 2026.

Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. Chasing the tail: Effective rubric-based reward modeling for large language model post-training. In *International Conference on Learning Representations*, volume 2026, pp. 133430–133457, 2026.

Keyu Zhao, Lingyan Kong, Fengli Xu, and Yong Li. Agentic-ideation: Sample efficient agentic trajectories synthesis for scientific ideation agents. *arXiv preprint arXiv:2606.31229*, 2026.

Appendices

A Reinforcement-learning objective and training details

Initialization and data. All training variants start from Qwen3.6-27B (Qwen Team, 2026). This dense model achieves an initial success rate of 41.02%, compared with 37.65% for Qwen3.6-35B-A3B under the same no-thinking evaluation setting. For PESTO, RL starts from the SFT checkpoint described in Section 3.3 and uses the designated RL half of the paper corpus. The RL-only ablation starts directly from Qwen3.6-27B and uses questions and rubrics from the full corpus. The 336,840 question-rubric records reported in Section 3.2 describe the full corpus, not the size of PESTO's RL portion.

Rewards and group-relative advantages. For each of $B = 32$ questions, we sample $G = 8$ responses at temperature 0.8. The online judge assigns each response a reward $R_i = R(x, y_i, r)$ equal to the fraction of audited rubric criteria it satisfies. We normalize rewards within each group:

$$\widehat{A}_i = \frac{R_i - \mathrm{mean}(R_1, \ldots, R_G)}{\mathrm{std}(R_1, \ldots, R_G) + \delta}, \qquad \delta = 10^{-6}. \tag{2}$$

CISPO objective and asynchronous training. We use an asynchronous implementation of CISPO (Chen et al., 2025). Let π_{train} denote the trainer policy immediately before an optimizer update, and let $\pi_{\mathrm{roll},i}$ denote the policy that generated response y_i. These can differ because generation and training use different engines and may use different policy versions. To account for this mismatch, we compute a detached, clipped importance weight for each response token:

$$w_{i,t} = \mathrm{sg}\left[\mathrm{clip}\left(\frac{\pi_{\mathrm{train}}(y_{i,t} \mid x, y_{i,<t})}{\pi_{\mathrm{roll},i}(y_{i,t} \mid x, y_{i,<t})}, 0.5, 2.0\right)\right], \tag{3}$$

where sg stops gradients. The lower floor prevents very small trainer-to-rollout probability ratios from receiving vanishing weights. For one question, the CISPO policy loss is

$$\mathcal{L}_{\mathrm{policy}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} w_{i,t}\widehat{A}_i \log \pi_\theta(y_{i,t} \mid x, y_{i,<t}). \tag{4}$$

The clipped importance weights are held fixed during the update, with gradients flowing through the log probabilities.

Each rollout batch contains $32 \times 8 = 256$ responses and is used for exactly one optimizer update, with the gradient evaluated at $\pi_\theta = \pi_{\mathrm{train}}$. We average the policy loss across questions and add a low-variance KL penalty to a frozen reference policy ($\beta = 0.005$) and an entropy bonus ($\eta = 10^{-4}$). The frozen KL reference is distinct from the per-update trainer policy π_{train}. Actor weights are broadcast after every update; rollout batches more than three policy versions behind the trainer are rejected and regenerated.

Training and judging protocol. We train with an FP32 LM head, thinking disabled, and a 16,384-token response budget. We use Adam with learning rate 10^{-6}, weight decay 0.1, and $(\beta_1, \beta_2) = (0.9, 0.98)$. A gpt-oss-20B judge supplies online rewards, while Gemini 3.1 Pro Preview performs held-out evaluation. Using different judges reduces dependence on the training judge but does not eliminate judge bias. During RL, the paper-derived reference answer is neither shown to the policy nor used as a reference-matching or embedding-similarity target; it influences learning only through the generated rubrics and their audited criteria. This restriction does not apply to the preceding SFT stage.

Computational cost and experimental scope. Our main 27B-model RL run comprises 10,000 optimizer updates and takes approximately two weeks on 16 GPU nodes. This cost made additional full-scale ablations of graph construction and rubric refinement impractical within our available compute budget. We therefore prioritize evaluating whether the complete training recipe transfers to execution-grounded research. We use an existing RL algorithm and do not claim that the individual data-construction components are necessary or optimal.

Table 5: **Reinforcement-learning configuration.**

RL algorithm	Asynchronous CISPO
Backbone	Qwen3.6-27B
LM head precision	FP32
`PESTO` RL initialization	SFT checkpoint
`PESTO` RL data	Designated RL half of the paper corpus
RL-only initialization / data	Qwen3.6-27B / full question-rubric corpus
Batch	32 questions $\times$ 8 responses; total 256
Updates per rollout batch	1
Sampling temperature	0.8
Thinking	Disabled
Response budget	16,384 tokens
Reward	Mean binary-rubric satisfaction
Advantage stabilizer	$\delta = 10^{-6}$
Optimizer	Adam, lr 10^{-6}, wd 0.1, betas $(0.9, 0.98)$
Regularization	KL 0.005; entropy 10^{-4}
Trainer/rollout importance-weight clip	$[0.5, 2.0]$ (Equation 3)
Maximum rollout staleness	3 policy versions
Weight broadcast	After every optimizer update
Online / evaluation judges	gpt-oss-20B / Gemini 3.1 Pro Preview

B CONSTRUCTING QUESTION–ANSWER PAIRS FROM PAPER GRAPHS

For each paper, let $G = (V, E)$ be its extracted DAG and $\rho(v)$ the semantic role of node v. An edge (u, v) indicates that v depends on u. A semantic pair (C, A) contains question-context nodes C and answer-evidence nodes A. Natural-language questions and answers are generated only after this selection. Node identifiers are retained to preserve provenance.

Selection rules. Each operation specifies input and output roles and a deterministic query using three constructors: a paper-level cut between two role-defined node sets; a target-local cut from role-compatible direct parents to a target; or a shared-parent cut from a context node to its compatible direct children. Every compatible rule is applied. Empty contexts or targets are discarded, and local rules use their specified paper-level fallbacks when no reliable local match is available. There is no random pair sampling or prescribed mixture of operation types.

Algorithm 1 summarizes these queries using a topological traversal. The traversal order is expository: membership of each output pair is determined by its query, rather than by the order in which nodes are visited. A paper-level query may be an unconditional rule or a fallback for targets left unmatched by local queries.

Standard research operations. The standard operations are direction-to-idea, literature-to-gap, problem/gap-to-method, method-to-experiments, results-to-analysis, method-to-critique, and critique-to-fix. These operations use the constructors at different granularities. Literature-to-gap pairs each gap with its prior-work parents and falls back to all prior-work chunks when no local pair exists. Method-to-experiments emits a combined paper-level pair and additionally pairs each setup with its directly dependent results. Results-to-analysis uses the result parents of each analysis, with all results as context for unmatched analyses. Problem/gap-to-method also emits component-level targets when multiple method-detail chunks are present, using problem, motivation, gap, and overview chunks as shared context.

Mixed-context research operations. Mixed context is the union of four explicit cross-stage rules:

1. **Background to method detail:** all background chunks form the context; the targets are method-detail chunks with at least one background chunk as a direct parent.
2. **Hypothesis to result:** each hypothesis is paired with its directly dependent result chunks.
3. **Results to limitations and future directions:** all result chunks are paired with the paper's limitation and future-direction chunks.

Algorithm 1 Dependency-based semantic-pair construction

Require: Paper DAG $G = (V, E)$, roles ρ, operation rules $\mathcal{K}$
Ensure: Semantic pairs $\mathcal{P}$ with operation labels and node identifiers

$\mathcal{P} \leftarrow \varnothing$
for each operation $k \in \mathcal{K}$ **do**
  $\mathcal{L}_k \leftarrow \varnothing$
  for each node v in a topological ordering of G **do**
    Add target-local matches at v specified by k to $\mathcal{L}_k$
    Add shared-parent matches rooted at v specified by k to $\mathcal{L}_k$
  end for
  $\mathcal{B}_k \leftarrow$ unconditional paper-level cuts specified by k
  $\mathcal{F}_k \leftarrow$ specified fallbacks for missing local matches
  for each $(C, A) \in \mathcal{L}_k \cup \mathcal{B}_k \cup \mathcal{F}_k$ **do**
    if $C \neq \varnothing$ and $A \neq \varnothing$ **then**
      Add (k, C, A) with source node identifiers to $\mathcal{P}$
    end if
  end for
end for
return $\mathcal{P}$

4. **Direction/gap to contribution:** problem, motivation, and gap chunks are paired with contribution summaries.

The first two rules require recorded dependencies. The latter two use paper-level context to support synthesis across multiple findings or goals, whose incoming dependencies may not be exhaustively recorded. Mixed-context pairs therefore expand the scenarios covered while retaining deterministic selection and source provenance.

From selected evidence to scored interactions. Each pair is naturalized into a question and one reference answer. The answer is converted into binary rubric criteria and audited using a panel of model responses. Criteria already producing mixed verdicts are preserved; all-pass criteria are made more specific where the reference supports it, and all-fail criteria are inspected for brittle requirements or valid headroom. Model identities are anonymized during revision. Programmatic checks reject references to panel responses and changes to protected criterion identifiers or aspects. Revised rubrics are evaluated against the same frozen responses. Appendix D provides the full graph, selected nodes, question, reference answer, and rubric for each worked example.

B.1 Answer multiplicity across related questions

We identify semantically related questions by embedding the first question variant for each unique paper–pair using `all-MiniLM-L6-v2`. We apply average-linkage agglomerative clustering with cosine distance separately within each research-operation type, using a distance threshold of 0.20. We count one reference-answer record per unique paper–pair represented in a cluster, so multiple question variants of the same pair do not increase the count. Averaging this count across the resulting clusters yields 2.18 reference-answer records per cluster. This statistic measures corpus-level answer multiplicity across related questions; each training record retains its own reference-derived rubric.

C EXAMPLE: FROM A PAPER GRAPH TO A QUESTION ANSWERING (QA) TASK

Task construction

KnowFormer (Liu et al., 2024) reviews path-based knowledge-graph reasoning and identifies two limitations: missing paths between entities and over-squashing of information from converging paths. Later passages evaluate robustness across path lengths and missing-fact ratios.
Figure 2 represents these passages as nodes, with directed edges linking the problem statement, prior work, identified gap, and downstream evaluation.
The `literature_to_gap` rule selects the blue prior-work node as question context and its direct child, the red gap node, as answer evidence. The context becomes a self-contained question about path-based reasoning limitations; the reference answer identifies missing paths and over-squashing. Experiment and analysis nodes show how the gap informs evaluation but are excluded from the question context.

Generated question

I'm trying to find a good research direction in knowledge graph reasoning. We know that standard embedding methods don't work well for inductive settings, so the community shifted toward path-based methods that look at relational paths and subgraphs between entities. But I suspect these path-based models have their own failure modes when dealing with complex graph topologies. Could you break down the specific limitations or gaps that occur when relying strictly on paths between head and tail entities?

Reference answer

Path-based methods for knowledge graph reasoning face two primary structural bottlenecks:

1. **Missing paths.** For a given query relation, there may be no existing path between the head and tail entities. When the graph is sparse or incomplete, path traversal fails because the required topological connection is absent.
2. **Over-squashing.** Even when paths exist, many paths may converge. Their information must then be aggregated and compressed into a fixed-size representation. This compression can discard fine-grained information needed for accurate reasoning.

A more robust reasoning method should therefore infer relations even when paths are missing and prevent information loss when aggregating many converging paths.

Binary rubric

The response is evaluated against the following criteria:

1. Does the response identify the absence of paths, or "missing paths," as a primary structural limitation?
2. Does the response identify over-squashing, or an information bottleneck during aggregation, as a primary structural limitation?
3. Does the response specify that the missing-path problem concerns the absence of a path between the head and tail entities for a query relation?
4. Does the response attribute missing paths to sparsity or incompleteness in the knowledge graph?
5. Does the response explain that over-squashing can arise when multiple paths converge?
6. Does the response describe over-squashing as the aggregation and compression of information from many paths into a single representation?
7. Does the response explain why path traversal fails when the necessary topological connection is absent?
8. Does the response explain that over-squashing can discard specific, fine-grained information needed for reasoning?
9. Does the response identify reasoning when paths are missing as a direction for developing more robust methods?
10. Does the response identify preventing information loss from dense, converging paths as a direction for developing more robust methods?

Example scoring. On this question, Claude Opus 4.8 satisfied all 10 rubric criteria, whereas o4-mini satisfied none. This example illustrates how the rubric distinguishes these two responses.

D MORE EXAMPLES OF (QA) TASKS

This appendix expands Section 3.2 with three examples. Each includes the full extracted semantic dependency DAG, naturalized question, reference answer, and audited binary rubric. Graphs show all extracted chunks and recorded intra-paper dependencies. Node fills indicate semantic roles; blue and red borders mark question context and answer evidence, respectively.

D.1 IDEATION FROM FLOW MATCHING GUIDE AND CODE

Full dependency DAG. Figure 8 shows all 24 extracted chunks from ***Flow Matching Guide and Code*** (Lipman et al., 2024) and their recorded dependencies. The graph captures a layered technical derivation.

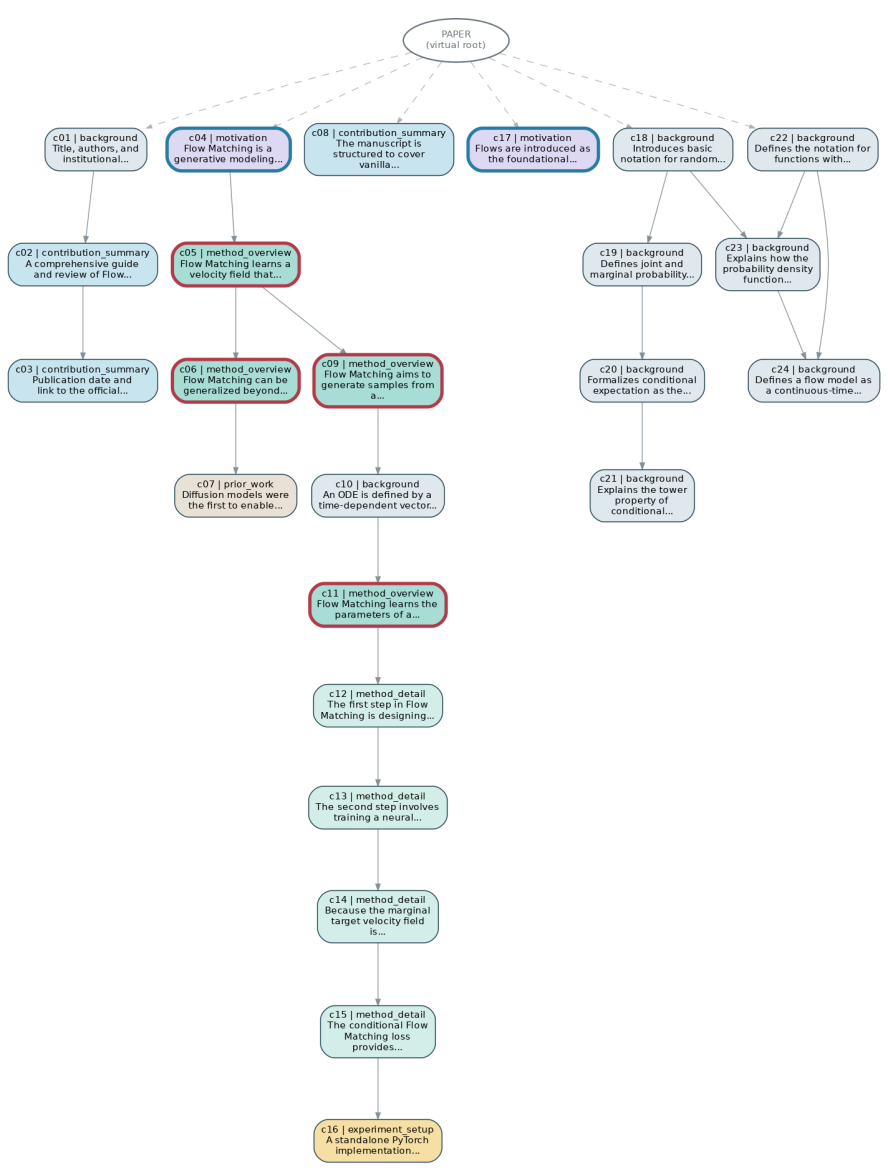

Figure 8: **Complete graph for *Flow Matching Guide and Code*.** All 24 extracted chunks and all recorded dependencies are shown. Blue borders identify the two motivation chunks used as question context; red borders identify the four method-overview chunks withheld as answer evidence.

Here are the naturalized question, reference answer, and audited rubric used for scoring.

Generated question

I'm working on generative modeling using deterministic flows, which transform a source distribution into a target distribution via ODEs. Older continuous normalizing flows required expensive ODE simulation and differentiation during training to maximize likelihood. What novel approach could we use to train these velocity fields without simulation, and how might this core idea be generalized to non-Euclidean domains like discrete state spaces?

Reference answer

To build a generative model, learn a velocity field that defines a flow ψ_t through an ordinary differential equation (ODE). This flow is a deterministic, time-continuous bijection on $\mathbb{R}^d$ that transforms a source sample $X_0 \sim p$ into $X_1 = \psi_1(X_0) \sim q$, where q is the target distribution.
Traditional continuous normalizing flows maximize training-data likelihood, requiring ODE simulation and differentiation through it during training. A simulation-free alternative first chooses a probability path p_t interpolating between $p_0 = p$ and $p_1 = q$, then trains a neural velocity field u_t^θ to implement this path by regressing instantaneous sample velocities. After training, generate samples by drawing $X_0 \sim p$ and solving the ODE defined by u_t^θ.
The framework extends beyond Euclidean spaces and deterministic flows. Continuous-time Markov chains (CTMCs) support discrete tasks such as language modeling, while flows on Riemannian manifolds support geometric applications such as protein modeling. More generally, continuous-time Markov processes (CTMPs) encompass diffusions and jump processes in continuous spaces, as well as discrete-state CTMCs. Across these settings, the recipe remains the same: choose a probability path from p to q on the state space $\mathcal{S}$, then train a generator that defines a Markov process implementing that path. The generator takes the role of the velocity field, unifying generative modeling across modalities and process types.

Binary rubric

The response is evaluated against the following criteria:

1. *key idea.* Does the response explain that the learned velocity field defines a flow through an ODE, and that generating the target requires transporting a source sample so that its endpoint matches the target distribution?
2. *key idea.* Does the response present the simulation-free training as a two-step recipe in which one first chooses a probability path interpolating source and target, and then trains a network to implement that specific path, contrasting this with likelihood training that differentiates through ODE simulation?
3. *technical detail.* Does the response describe a probability path connecting the source and target distributions, with enough detail to explain what the velocity field is meant to implement?
4. *technical detail.* Does the response explain how a regression objective can train the velocity field using velocity targets available from the chosen path, without differentiating through ODE simulation?
5. *justification.* Does the response distinguish simulation-free training from generation, which may require integrating the learned dynamics?
6. *key idea.* Does the response generalize the framework by replacing the velocity field with a generator that defines a Continuous Time Markov Process implementing a chosen path, while preserving the same two-step recipe (choose a path on the relevant state space, then train the generator)?
7. *technical detail.* Does the response explain how dynamics on a discrete state space could play the role that a velocity field plays in continuous space, for example through transition rates of a continuous-time Markov chain?
8. *technical detail.* Does the response explain how the path-and-dynamics recipe could be adapted to at least one non-Euclidean state space, beyond simply naming that space?
9. *specifics.* Does the response provide concrete application examples for the non-Euclidean domains, such as language modeling for discrete spaces or protein folding/chemistry for manifolds?
10. *justification.* Does the response explain why separating the choice of probability path from the dynamics that implement it makes the approach extensible to other processes or data types?

D.2 CRITIQUE-TO-FIX FROM MINICTX

Full dependency DAG. Figure 9 shows all 57 extracted chunks from ***miniCTX:Neural Theorem Proving with (Long-)Context*** (Hu et al., 2025) and their recorded dependencies. The graph has a hub-dominated structure around the benchmark definition.

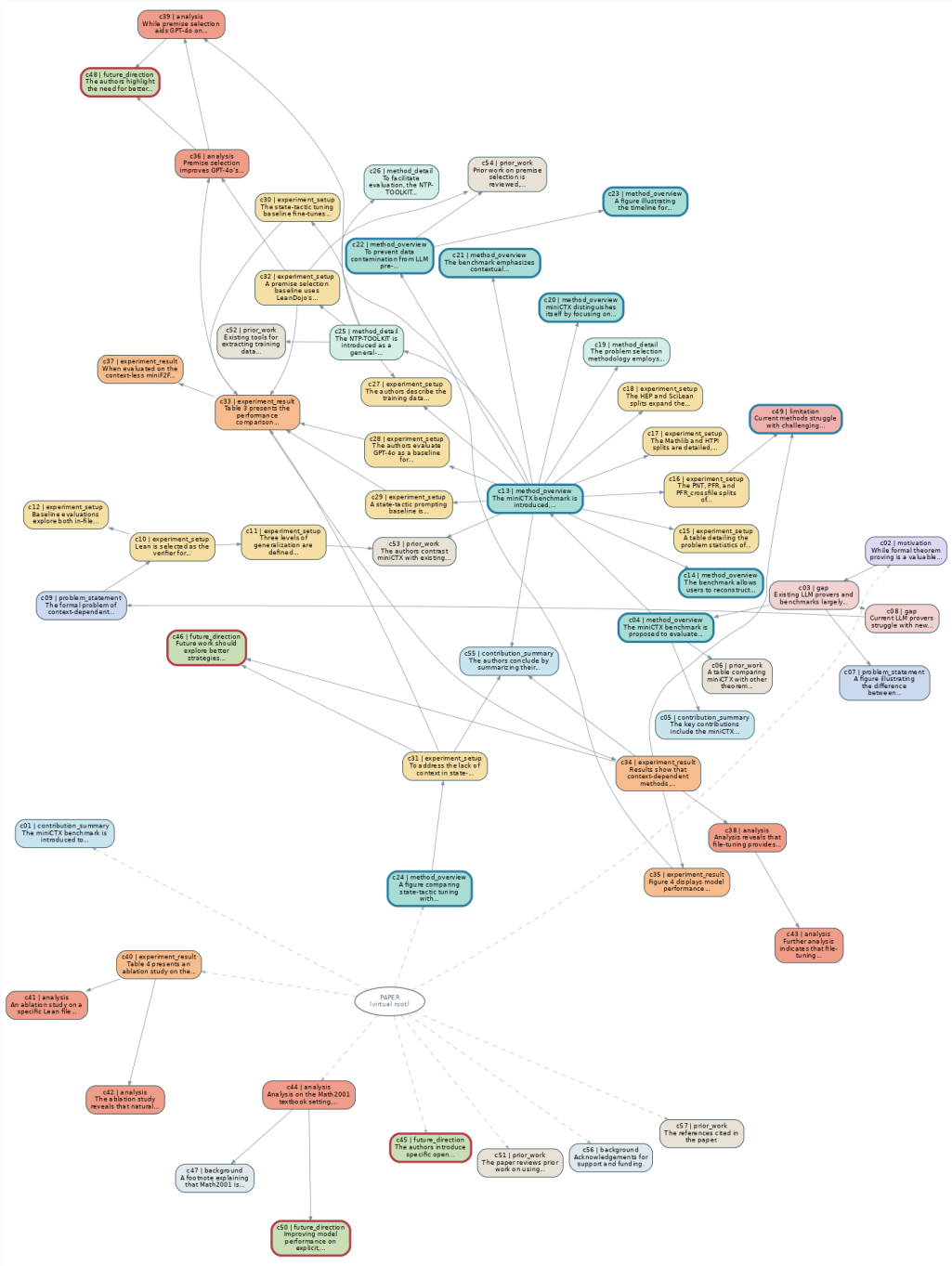

Figure 9: **Complete graph for *miniCTX*.** All 57 chunks and their dependencies form a radial, hub-dominated structure around the benchmark definition. The limitation-and-method context used to propose a repair is outlined in blue; the selected forward-looking targets are outlined in red.

Here are the naturalized question, reference answer, and audited rubric used for scoring.

Generated question

We are evaluating neural theorem provers on a new Lean 4 benchmark that includes complex, real-world projects and requires using both in-file and cross-file context. However, our current baselines have several distinct weaknesses: our file-tuning method truncates context to a 1024-token budget which discards vital information, our attempts to use premise selection for cross-file dependencies don't consistently improve performance, and the models perform poorly when powerful automation tactics are banned and explicit step-by-step proofs are required. What specific future research directions or methodological fixes could address these open challenges?

Reference answer

To address these open challenges and improve performance on the benchmark, you could explore several specific future directions: Making better use of long contexts. Consider moving beyond the current file-tuning method that simply truncates contexts to a 1024-token budget, as this discards highly useful contextual information. While providing a model like GPT-4o with 8,000 tokens of context shows gains compared to providing no context at all, its absolute performance remains low. For future work, you could explore strategies such as feeding in the entire context, implementing retrieval mechanisms, or using a mixture of both approaches. Repository-level context. While current evaluations focus heavily on in-file context, many problems require context from outside the current file. Since incorporating premise selection to leverage cross-file context does not consistently improve performance—even on datasets with high cross-file dependencies—there is a clear need to investigate how to better integrate premise selection with in-file context. You can utilize the benchmark's comprehensive metadata to reconstruct the entire environment, which will allow for a much deeper investigation into premise selection and other potential methods for leveraging cross-file context. Working with constraints, model performance drops significantly when proofs cannot use powerful automation tactics. Because models have a strong tendency to invoke these powerful tactics, they struggle when forced to generate more explicit, step-by-step proofs. Improving model performance in these constrained settings is an interesting and valuable direction for future research.

Binary rubric

The response is evaluated against the following criteria:

1. *key idea.* Does the response propose a specific alternative to fixed truncation that names at least two distinct strategies for making better use of long context, such as feeding in the entire/full context and using retrieval, or a mixture of both?
2. *key idea.* Does the response identify that premise selection for cross-file dependencies fails to consistently help and propose specifically investigating how to better integrate premise selection with in-file context (rather than treating cross-file and in-file context as separate problems)?
3. *key idea.* Does the response attribute the difficulty of the constrained setting to models' reliance on powerful automation tactics and propose a direction that specifically targets generating explicit, step-by-step proofs when those tactics are unavailable?
4. *justification.* Does the response justify the need for better long-context handling by noting that current truncation discards useful information, or that simply increasing context size still yields low absolute performance?
5. *justification.* Does the response explain that models struggle with explicit, step-by-step proofs because they have a strong tendency or bias to invoke powerful automation tactics?
6. *technical detail.* Does the response suggest feeding the entire context into the model as a potential solution for long contexts?
7. *technical detail.* Does the response suggest implementing retrieval mechanisms (or a mixture of retrieval and full context) to handle long contexts?
8. *technical detail.* Does the response suggest investigating the integration of premise selection specifically with in-file context?
9. *technical detail.* Does the response propose a way to recover or use relevant cross-file project context, and explain how that context would be made available to the prover?
10. *specifics.* Does the response propose evaluating whether its context-handling approach improves proof success under different context budgets or cross-file dependency levels?

D.3 BENCHMARK DESIGN FROM BAG OF TRICKS

Full dependency DAG. Figure 10 shows all 32 extracted chunks from ***Bag of Tricks: Benchmarking of Jailbreak Attacks on LLMs*** and their recorded dependencies. A common evaluation setup branches into eight factor-specific results, which converge into aggregate configurations and analysis.

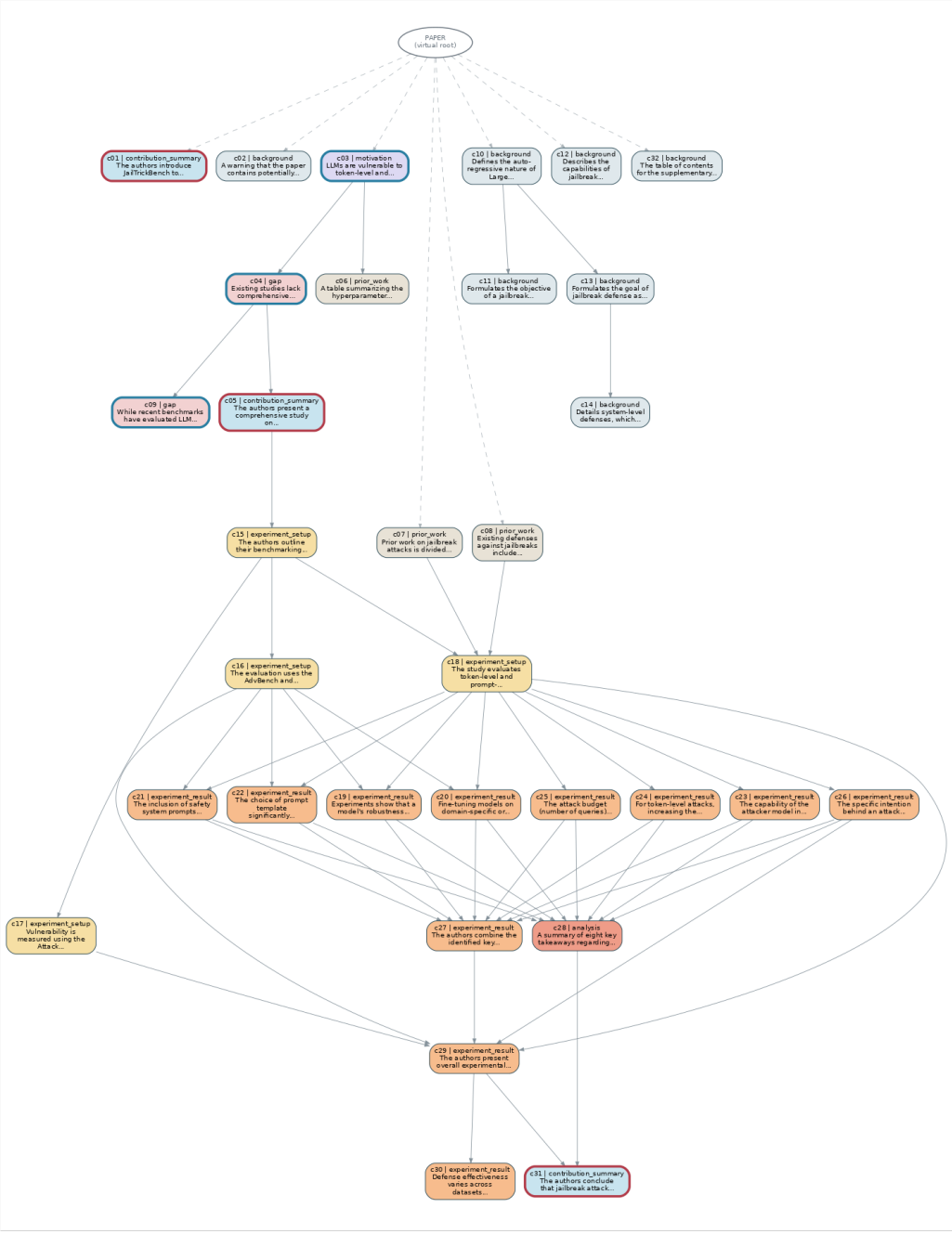

Figure 10: **Complete graph for *Bag of Tricks*.** The 32-chunk graph has a fork–join structure: a common evaluation setup produces eight factor-specific results, which converge into aggregate configurations and analysis. Blue borders mark the problem and gap context selected for a benchmark-design question; red borders mark the withheld contribution summaries used as answer evidence.

Here are the naturalized question, reference answer, and audited rubric used for scoring.

Generated question

Current jailbreak benchmarks for LLMs, like JailbreakBench, mostly focus on original, undefended models and overlook how specific attack settings (like attacker budget) or target model factors impact vulnerabilities. How could we design a comprehensive benchmarking study that systematically evaluates these missing target-level and attacker-level factors across defense-enhanced LLMs?

Reference answer

To address current evaluation limitations, develop a standardized benchmark for jailbreak attacks on defense-enhanced LLMs. Evaluate token-level and prompt-level attacks across eight factors covering target-model settings, such as model size, safety alignment, system prompts, and chat templates, and attacker settings, such as budget and suffix length.
Establish a baseline by testing seven representative attacks against six defense methods on AdvBench and MaliciousInstruct. The proposed scale comprises approximately 354 experiments and 55,000 GPU hours on A800-80G hardware.
The study could reveal how evaluation settings affect apparent robustness. Model size may not reliably predict robustness; fine-tuning can alter safety alignment; safe system prompts can strengthen defenses; and unsuitable chat templates can distort vulnerability assessments. Testing combinations of factors would also show how improper settings degrade token-level and prompt-level attack performance.
Together, these experiments would provide defense-enhanced baselines and a practical "bag of tricks" for understanding attack limitations. By documenting sensitivity to model configuration and attacker resources, the framework would support reproducible comparisons, guide future research, and motivate continuous, standardized evaluation of jailbreak risks.

Binary rubric

The response is evaluated against the following criteria:

1. *key idea.* Does the response frame the framework as standardizing evaluation of jailbreak attacks specifically against defense-enhanced LLMs, explicitly treating the defense method as an evaluated dimension rather than testing only undefended base models?
2. *key idea.* Does the response explicitly state that the framework should investigate both token-level and prompt-level attacks?
3. *technical detail.* Does the response identify multiple distinct target-model-level factors — spanning at least three of model size, safety alignment/fine-tuning, system prompt configuration, and chat template type — as explicit dimensions to be varied?
4. *technical detail.* Does the response identify attacker-level factors that include both the attacker's resource budget (e.g., query or compute budget) and the adversarial suffix/prompt length as dimensions to be varied?
5. *technical detail.* Does the response propose an experimental setup that crosses multiple representative jailbreak attacks with multiple distinct defense methods as a factorial matrix?
6. *justification.* Does the response articulate a non-obvious expected finding about how basic model settings affect robustness — such as robustness not strictly correlating with model size, fine-tuning altering original safety alignment, or safe system prompts significantly enhancing robustness?
7. *justification.* Does the response test interactions among attacker settings, rather than assuming that each setting affects attack performance independently?
8. *justification.* Does the response justify the experiments by stating they will establish a baseline benchmark for attack performance on actively defended models?
9. *specifics.* Does the response specify suitable, reproducible evaluation prompts or datasets and explain how they would be used consistently across conditions?
10. *specifics.* Does the response describe a feasible evaluation scale or sampling plan sufficient to compare the proposed factors?

E NANOGPT AND GRPO: RESEARCH TOWARD A FIXED OBJECTIVE (FO)

E.1 EVALUATION PROTOCOL

We evaluate eight independent proposals per model and task, each generated from the initial task context without access to other proposals or their outcomes. Proposal indices follow the report's zero-based numbering and do not indicate feedback rounds. We report the mean and best scores across the eight candidates.

nanoGPT: task, training, and scoring

Task. Minimize held-out FineWeb validation cross-entropy by modifying only `train.py` in the nanoGPT environment.
Model and data. Train a GPT-2-style model from scratch with 12 layers, 12 attention heads, embedding width 768, vocabulary size 50,257, and context length 1,024, using GPT-2-tokenized FineWeb10B shards.
Training defaults. AdamW with learning rate 0.0015, 256 warmup iterations, weight decay 0.1, gradient clipping at 1.0, and a global batch of 262,144 tokens. Proposals may modify permitted aspects of training.
Evaluation. Compute cross-entropy over exactly 10,485,760 validation tokens. The validation loss, token budget, evaluation frequency, and autoregressive causality are fixed. The score is the minimum reported `val_loss` from a single final evaluation. Lower is better.

GRPO: task, training, and scoring

Task. Maximize strict held-out MATH accuracy by modifying only `grpo.py`, `grpo_utils.py`, and `utils.py` in the GRPO environment.
Model and data. Post-train `Qwen/Qwen2.5-Math-1.5B` on 7,500 MATH training problems, with 500 held-out test problems, using `prompts/r1_zero.prompt`.
Training defaults. Train for 20 GRPO epochs with learning rate 10^{-5}. Each epoch samples 128 training prompts and generates eight responses per prompt, followed by one update pass. Use microbatch size 2, gradient accumulation 32, and `grpo_clip` loss with clipping range 0.2. Proposals may modify permitted aspects of training.
Generation and rewards. Sample with temperature and top-p both 1.0, generating at most 1,024 tokens and stopping after `</answer>`, including the stop string. The default reward is 1 only for a mathematically correct answer with the required `</think> <answer> ... </answer>` structure, and 0 otherwise. Training rewards may change; held-out grading is fixed.
Evaluation. Evaluate all 500 held-out problems every two epochs using the fixed evaluator and strict answer grader. Higher accuracy is better.

E.2 IDEA-GENERATION PROMPTS

The idea model receives the system message and user-message template below. The system message and task constraints are reproduced verbatim; the template uses placeholders for run-specific code and constraints.

Idea generation: exact system message

You are an expert ML research scientist. Propose one simple, novel, generalizable experiment that a strong engineer can implement in the supplied codebase. Focus on an algorithm, architecture, objective, optimizer, curriculum, or regularization mechanism rather than broad hyperparameter tuning. Your task is idea generation only; do not write a full patch.

Parallel idea generation: user-message template

Below are the relevant code files for the <environment> research environment:
<deterministically generated, line-numbered code context>
<task-specific constraints>
Return exactly one experiment and no other text, using this format: [Experiment] concise description and scientific motivation [Code Changes] precise implementation steps and any fixed hyperparameter values [End]

For each environment (`nanogpt` or `grpo`), the code context includes line-numbered `.py` and `.sh` files in sorted order, excluding `fineweb.py` and `run.sh`. The contents of `evaluate.py` are replaced with "Reserved evaluation code; it may not be read or modified."

nanoGPT: exact task constraints

Propose exactly one experiment. Do not modify evaluation data, evaluation metrics, evaluation frequency, hard runtime limits, or W&B identity. External dependencies are unavailable. Do not alter the validation cross-entropy expression, val_loss_every, val_tokens, or autoregressive causality; never expose future tokens.

GRPO: exact task constraints

Propose exactly one experiment. Do not modify evaluation data, evaluation metrics, evaluation frequency, hard runtime limits, or W&B identity. External dependencies are unavailable. If changing rewards, add a training-only reward and preserve the original evaluation grader and held-out evaluation path.

The parser requires one experiment with experiment and implementation sections, accepting unambiguous Markdown headings as equivalent tags.

E.3 Implementation and execution workflow

The idea model proposes an experiment without editing code. A fixed implementer, Codex CLI `0.149.1` with `gpt-5.6-sol` at `xhigh` reasoning effort, receives the stored `idea.json` `idea` field and edits a fresh copy of the clean environment. It runs in a session subject to the restrictions below.

Codex: implementation-prompt template

Binding research idea
<exact idea.json idea field>
#Implementation contract
Implement exactly the idea above in this <environment> environment. You are the implementer, not another ideator: do not replace, broaden, or combine the idea. You may edit only: <editable files>.
Do not create source files, install dependencies, use the network, modify evaluation data or metrics, alter validation frequency, change the hard runtime limit, change W&B identity, or run full training. Preserve autoregressive causality. For GRPO, training-only reward changes are allowed but evaluation.py, drgrpo_grader.py, MATH, and the held-out evaluator are immutable. The GRPO validator is deliberately textual: do not add, remove, or edit any existing source line containing 'eval_', 'evaluate_model', 'evaluate_vllm', 'r1_zero_reward_fn_eval', or 'eval_dataset_path', even when the intended change is training-only.
Inspect the repository, make the implementation directly, and run only lightweight syntax/static checks. Leave the working tree containing the completed implementation.

For nanoGPT, the editable-files placeholder is `train.py`; for GRPO, it is `grpo.py`, `grpo_utils.py`, `utils.py`. The generator can see `drgrpo_grader.py`, but the implementer cannot edit it. A proposal referring to that file must be realized through the permitted training files or fail validation. Consequently, the raw proposal and executed patch may differ in implementation location while preserving the proposed mechanism.

E.4 Per-proposal results and qualitative diversity

Tables 6 and 7 report scores and summarize the proposed mechanisms. Bold scores mark the best proposal within each model and task. The summaries describe proposed or reported implemented mechanisms.

Table 6: nanoGPT: all eight reported parallel proposals per idea model. Lower loss is better.

ID	Model	Proposal	Loss
0	Base	**Adaptive per-layer gradient clipping.** After backpropagation, calculate a separate gradient norm for every Transformer block and clip each block relative to its own empirical scale. The objective is to restrain unstable layers without shrinking useful updates in stable layers.	3.2537
1	Base	**Stochastic depth for residual branches.** Randomly drop complete attention or MLP residual branches during training and rescale surviving branches to preserve their expectation. Inference uses the full network.	4.0317
2	Base	**Depth-dependent learning rates.** Divide parameters into optimizer groups by Transformer depth and assign smaller learning rates to deeper blocks. This is a static layer-wise schedule intended to stabilize sensitive upper layers.	3.2543
3	Base	**Layer-wise learning-rate decay.** Apply a geometric learning-rate schedule across Transformer blocks through AdamW parameter groups. It is very similar to Base round 2 and mainly differs in the exact decay formula.	3.2582
4	Base	**Adaptive block-wise gradient normalization.** After backward, calculate one L2 gradient norm per Transformer block and divide that block's gradients by its norm before the existing global clipping and AdamW step. This equalizes block contributions, though it also removes potentially meaningful gradient-magnitude differences.	**3.2532**
5	Base	**Per-layer RMS gradient scaling.** Divide every block's gradients by their RMS magnitude before the optimizer update. This equalizes update scales across depth instead of only clipping extreme gradients.	3.3712
6	Base	**Learnable backward-only gradient scaling.** Add one log-scale per layer and a custom autograd operation that leaves forward activations unchanged but multiplies the backward gradient by exp(scale). The proposal intends to learn which layers need larger updates, although its proposed custom backward does not clearly provide a gradient to the scale itself.	3.8230
7	Base	**Adaptive residual scaling.** Add per-block factors controlling the balance between the identity path and learned residual transformations. This seeks to keep activation and gradient magnitudes stable across depth.	3.2991
0	PESTO	**Progressive context-length curriculum.** Begin training on short sequences, such as 128 tokens, and gradually increase toward the full 1024-token context. This lowers early optimization difficulty and may permit more updates per unit of compute before long-range dependencies are introduced.	3.2735
1	PESTO	**Self-conditioned attention dropout.** Derive the dropout probability from attention-logit magnitude or concentration instead of using one uniform rate. Highly concentrated and uncertain attention regions therefore receive different amounts of regularization.	3.3374
2	PESTO	**Progressive skip/residual scaling.** Multiply attention and MLP residual updates by a fixed factor that decreases with layer depth. Deep blocks remain closer to the identity path and contribute smaller perturbations to the accumulated residual stream.	3.2692
3	PESTO	**Learnable residual gates.** Give each block a trainable scalar controlling how much of its attention or MLP output enters the residual stream. Initialize the gates conservatively and allow training to discover the useful branch strength at each depth.	3.2502
4	PESTO	**Layer-wise stochastic depth.** Randomly remove complete attention or MLP residual updates, usually with a drop probability that increases with depth, and rescale surviving paths. This trains an implicit ensemble of shallower subnetworks while retaining the complete model at evaluation.	3.2683
5	PESTO	**Inverse-depth learning-rate scaling.** Construct optimizer parameter groups with a geometric learning-rate multiplier by layer. Lower and upper blocks adapt at different speeds while the global learning-rate schedule remains unchanged.	3.2541
6	PESTO	**Layer-dependent attention dropout.** Use stronger attention dropout in shallow layers and weaker dropout in deep layers. The hypothesis is that shallow lexical patterns need more regularization, whereas deep semantic and long-range dependencies should be preserved.	3.2631
7	PESTO	**Fixed-base RoPE.** Remove the absolute-position table, precompute sine/cosine frequencies, and rotate attention queries and keys before causal attention. Update normal and cached decoding while preserving the original validation loss and causal mask.	**3.2373**

Table 7: GRPO: all eight reported parallel proposals per idea model. Higher reward is better.

ID	Model	Proposal	Reward
0	Base	**Inverse-square-root length advantage.** Compute ordinary group-relative advantages and multiply each response's advantage by $1/\sqrt{\text{length}}$. This reduces the influence of long rollouts without changing their binary correctness reward.	0.4460
1	Base	**Length-normalized group reward.** Divide the binary reward by $1 + \alpha \times$ length, then compute group means, standard deviations, and advantages from the adjusted values. Short correct responses rank above long correct responses.	0.3620
2	Base	**Median-relative length scaling.** Multiply a standard advantage by group_median_length / response_length. Responses shorter than the group median receive stronger updates and longer responses are damped.	0.4160
3	Base	**Answer-correctness-conditional proximal regularization.** Pass a correctness mask into the loss and add a small squared current-versus-old log-probability penalty only for incorrect rollouts. Correct responses retain the ordinary clipped GRPO objective, while failed trajectories are discouraged from drifting into arbitrary behavior.	**0.4760**
4	Base	**Entropy regularization for uniform failures.** The proposal begins as group-diversity shaping but settles on adding token entropy to the policy loss. This supplies an exploration signal when every rollout in a group is identical or incorrect.	0.4720
5	Base	**Length-normalized correctness reward.** Wrap the training grader so correct answers receive reward divided by a function of response length while incorrect answers remain zero. The evaluation reward is untouched.	0.4060
6	Base	**Square-root length-normalized reward.** Divide a correct response's reward by $\sqrt{\text{length}}$ and use it for group advantages while retaining binary evaluation. This is a softer version of Base round 5.	0.3640
7	Base	**Linear token-efficiency penalty.** Subtract a fixed cost per generated token from the training reward. This directly favors the shortest possible reasoning and produced the weakest Base result.	0.1160
0	PESTO	**Length-penalized reward shaping.** Subtract a response-length cost from the binary training reward before group normalization. Correctness evaluation remains unchanged, so the proposal tests whether concision during training improves held-out accuracy.	0.4240
1	PESTO	**Variance-normalized policy contribution.** Divide the advantage by $\sqrt{\text{response length}}$ inside the policy loss. Because accumulated token log-probability variance grows with response length, this is intended to make updates from short and long responses comparable.	0.4740
2	PESTO	**Think–verify reward shaping.** Replace a purely binary training signal with multiple training-only components that reward final correctness, valid answer format, and evidence of self-checking or correction. The proposal attempts to give partial credit to useful reasoning behavior while retaining strict final-answer evaluation.	0.4220
3	PESTO	**Length-normalized entropy regularization.** Add a token-entropy bonus, average it over valid response tokens so strength is independent of response length, and decay the entropy coefficient during training. This encourages broad exploration early and greater exploitation later.	0.4660
4	PESTO	**Length-normalized reward before advantage calculation.** Divide each raw binary training reward by generated-token count, then compute the usual group mean, standard deviation, and advantage from the adjusted rewards. Response lengths are logged and the strict held-out evaluator remains unchanged.	**0.4840**
5	PESTO	**Entropy bonus with temperature annealing.** Add masked token entropy to the GRPO loss while gradually reducing rollout sampling temperature. Early training explores diverse solutions; later training concentrates probability on successful reasoning paths.	0.4780
6	PESTO	**Length-aware tie-breaking and advantage scaling.** When all rollouts in a group share the same correctness reward, use relative response length to produce a nonzero ranking signal. Then damp each resulting advantage according to length.	0.4760
7	PESTO	**Step-weighted policy gradient.** Convert the response mask into normalized token positions and apply larger advantage weights to early reasoning tokens, decaying toward later tokens. The hypothesis is that early reasoning mistakes have greater downstream impact and deserve stronger credit assignment.	0.4400

E.5 HIGHEST-SCORING PROPOSAL RESPONSES

The following boxes show the response for each model's best proposal. Note that the excerpts reproduce model-generated proposals and may contain inaccurate motivations; downstream scores evaluate the implemented changes rather than independently validating every explanatory claim.

nanoGPT: base model, proposal 4, loss 3.2532

[Experiment] **Adaptive Layer-Specific Learning Rates via Gradient Norm Proxies** Standard AdamW applies a single global learning rate to all layers, ignoring the fact that early and late layers in Transformers often operate at different optimization scales and convergence speeds. This experiment implements a lightweight, zero-overhead regularization/optimizer modification that scales the learning rate for each block's parameters inversely proportional to the magnitude of their gradients. This acts as a dynamic, per-layer learning rate scheduler that stabilizes training for sensitive layers (typically early attention blocks) while allowing aggressive updates for robust layers (typically late MLP blocks), often leading to faster convergence and lower final loss without tuning the global LR.
[Code Changes] ...

nanoGPT: `PESTO`, proposal 7, loss 3.2373

[Experiment] Rotary Position Embeddings (RoPE) for Improved Length Generalization
Scientific Motivation The current 'GPT' model uses absolute sinusoidal position embeddings ('wpe'). Absolute embeddings struggle to generalize to sequence lengths longer than those seen during training because the model learns specific associations between position indices and token representations. **Rotary Position Embeddings (RoPE)** encode relative positions by rotating query and key vectors in 2D subspaces before attention. This injects relative positional information directly into the attention scores, making the model inherently invariant to absolute position shifts and significantly improving extrapolation to longer contexts. RoPE is standard in modern LLMs (Llama, Mistral) and often outperforms absolute embeddings in downstream tasks requiring structural reasoning or long-context retention, without increasing parameter count.
Hypothesis Replacing absolute position embeddings with RoPE will:
1. Improve validation perplexity on the fixed 1024-length sequences due to better relative position modeling.
2. Provide a more robust representation that generalizes better if the model is later tested on longer sequences (though evaluation remains at 1024).
3. Maintain or slightly improve training speed, as RoPE is a lightweight element-wise operation compared to the embedding lookup, and removes the need for a separate position embedding table.
Constraints & Rules - **No external dependencies**: RoPE is implemented using basic PyTorch operations ('torch.cos', 'torch.sin', 'torch.cat', etc.).
- **No evaluation changes**: 'val_loss_every', 'val_tokens', and the evaluation loop remain identical. The loss is still computed over the standard causal next-token prediction.
- **No architectural bloat**: RoPE replaces 'wpe' entirely. The total parameter count decreases slightly (removing 'block_size * n_embd' parameters), keeping the model size comparable or smaller.
- **Causality preserved**: RoPE does not alter the causal mask ('is_causal=True' in 'scaled_dot_product_attention').
[Code Changes] ...

GRPO: base model, proposal 3, reward 0.4760

[Experiment] Implement "Answer-Correctness-Conditional KL Regularization" (AC-KL). Standard GRPO often suffers from high variance and policy collapse because incorrect rollouts (reward=0) receive zero or negative advantages, providing no gradient signal to push the policy away from bad behaviors other than via the relative ranking within the group. By introducing a small, constant penalty proportional to the KL divergence from the reference policy *only* for rollouts that are factually incorrect (answer_reward=0), we regularize the policy against diverging into arbitrary hallucinations when it fails. This is distinct from standard KL regularization

because it applies pressure only when the model is wrong, preserving the exploration incentive for correct answers while preventing the policy from exploiting formatting hacks or collapsing to low-probability gibberish that accidentally gets grouped with correct answers. It is simple to implement by modifying the loss function in ‘grpo_utils.py‘ and requires no new external dependencies.
[Code Changes] ...

GRPO: PESTO, proposal 4, reward 0.4840

[Experiment]
Experiment: Length-Normalized Advantage Scaling for GRPO in Math Reasoning
Scientific Motivation Current Group Relative Policy Optimization (GRPO) implementations normalize rewards by the group mean and standard deviation. However, in math reasoning tasks, longer chain-of-thought responses naturally accumulate higher token-level log-probability penalties and often receive higher absolute rewards if correct, creating a confounding signal where the model may prefer verbose, low-confidence reasoning over concise, high-confidence solutions. This experiment introduces **Length-Normalized Advantage Scaling (LN-AS)**. The hypothesis is that by normalizing the raw reward by the sequence length of the response before computing group-relative advantages, we penalize unnecessary verbosity and encourage efficient reasoning. This acts as a regularizer against "reasoning bloat," potentially improving the signal-to-noise ratio in the policy gradient and leading to faster convergence or higher accuracy on complex problems.
[Code Changes] ...

F AI ENGRAM: OPEN-ENDED PAPER EXTENSION (OE)

In this section, we compare how Qwen3.6-27B and PESTO develop an extension of AI Engram (Kwon et al., 2026). Each model independently reads the same paper and official code context, identifies a gap, and carries its selected decisions forward through idea generation, method development, experimental design, and analysis. Both identify the same gap, but arrive at corrections with opposite relationships to their proposed selectivity measure.

Evaluation pipeline. At each research-decision stage, the advising model proposes **16 candidate next steps**. A fixed Codex implementer selects the most promising candidate, which becomes context for the next stage. This process is run independently for each model. After selecting the method and experimental plan, the implementer executes the experiment and returns the results to the same advising model. The model then proposes 16 analyses and next steps; the implementer selects one bounded revision, applies it, and evaluates the revised method. The advising model subsequently generates 16 final analyses, from which the implementer selects one. We repeat the complete workflow eight times for each advising model. The reported trajectory for each model is selected by the final Overall score, applied identically to both models. All reported metrics and qualitative analyses correspond to that same selected trajectory.

Experimental specification. We evaluate the selected extension using the pinned TOFU benchmark, comprising 400 forget, 400 retain, 400 holdout, 100 real-author, and 117 world-fact examples. We use 4,000 full-set examples to construct the reference covariance. Experiments use the pinned Llama-3.2-1B TOFU full and retain-90 models and the official AI Engram implementation, with $\alpha = 1$, answer-token masking, and all supported linear layers.

Success criteria. Success requires Overall ≥ 0.79369885, Memorization ≥ 0.93929008, Utility ≥ 0.90350547, EM ≤ 0.04229214, and Privacy ≥ 0.66036905. It additionally requires a Privacy gain of at least 0.02 over the reproduced adaptive Engram baseline, with a positive lower bound on the paired 95% bootstrap confidence interval for that gain. The selected centered-selectivity correction satisfies all prespecified criteria.

Uncertainty estimation. The reported 95% confidence interval for the Privacy gain, $[+0.0401, +0.0621]$, is computed using 10,000 paired row-bootstrap resamples of the 400 fixed

evaluation examples used for the Privacy comparison. This interval quantifies evaluation-row uncertainty conditional on the selected trajectory, rather than variability across independently generated research trajectories.

Selected research decisions. Table 8 summarizes the selected responses at each stage. Both models use a forget-to-retain projection-norm ratio as a selectivity proxy. The difference is how this ratio controls the correction: Qwen3.6-27B decreases the correction as selectivity increases, whereas `PESTO` increases it. The experimental plans share the same controls and success criteria, but test these different corrections. Their results lead to different diagnoses and revisions.

Table 8: **Selected decisions and their consequences.** Decision rows paraphrase the response selected from each model's 16 candidates; experimental rows report measured outcomes. Both models independently identify the same gap, but develop corrections with opposite relationships to selectivity, preserved after revision.

Stage	**Qwen3.6-27B**	**PESTO**
Literature → Gap	Identifies that projection magnitude measures edit size without distinguishing forget-specific from retention-relevant information.	Independently identifies the same gap: correction strength should reflect forget-versus-retain selectivity.
Gap → Idea	Proposes *Selectivity-Gated Projection Scaling*: protect general-purpose layers using an inverse gate on the forget-to-retain projection ratio.	Proposes *Selectivity-Gated Adaptive Scaling*: amplify corrections in layers with higher forget-to-retain projection ratios.
Idea → Method	Sets $c_\ell = u_\ell = (1 + 0.5S_\ell)^{-1}$. Recognizes that higher selectivity yields smaller corrections, contradicting its intent, but retains the equation.	Sets $c_\ell = \mathrm{clip}(s_\ell, 0.1, 5.0)$, giving larger corrections to more selective layers, consistent with its intent.
Method → Experiments	Tests inverse gating against adaptive Engram and $c_\ell = 1$. Requires Privacy gain ≥ 0.02, a positive confidence-interval lower bound, and preservation bounds.	Tests increasing corrections against the same baseline and identity control, with identical Privacy, confidence-interval, and preservation criteria.
Initial experiment	Mean factor ≈ 0.656 suppresses edits. Privacy falls to 0.0619; EM rises to 0.6207.	Mean factor ≈ 1.05 amplifies edits. Privacy rises to 0.7747, but EM of 0.1093 violates its bound.
Results → Analysis and revision	Diagnoses global dampening; sets $c_\ell = u_\ell / \bar{u}$. Restores mean one while retaining decreasing corrections with selectivity.	Diagnoses global amplification; sets $c_\ell = \mathrm{clip}(s_\ell / \bar{s}, 0.1, 5.0)$. Normalizes selectivity before clipping while preserving increasing corrections with selectivity.
Final evaluation	Recovers, but remains below adaptive Engram in Overall (0.8014 vs. 0.8137) and Privacy (0.6176 vs. 0.6404).	Improves Overall to 0.8396, Privacy to 0.6915, and EM to 0.0191, while meeting Memorization and Utility preservation bounds.