The Automated LLM Speedrunning Benchmark: Reproducing NanoGPT Improvements

Bingchen Zhao[⋆1,2] **Despoina Magka**[⋆1] **Minqi Jiang**[⋆1]
Xian Li[1] **Roberta Raileanu**[1] **Tatiana Shavrina**[1] **Jean-Christophe Gagnon-Audet**[1]
Kelvin Niu[1] **Shagun Sodhani**[1] **Michael Shvartsman**[1] **Andrei Lupu**[1]
Alisia Lupidi[1] **Edan Toledo**[1] **Karen Hambardzumyan**[1] **Martin Josifoski**[1]
Thomas Foster[1] **Lucia Cipolina-Kun**[1] **Abhishek Charnalia**[1] **Derek Dunfield**[1]
Alexander H. Miller[1] **Oisin Mac Aodha**[2] **Jakob Foerster**[1] **Yoram Bachrach**[1]

[⋆]Equal contribution

[1]Meta Superintelligence Labs [2]University of Edinburgh

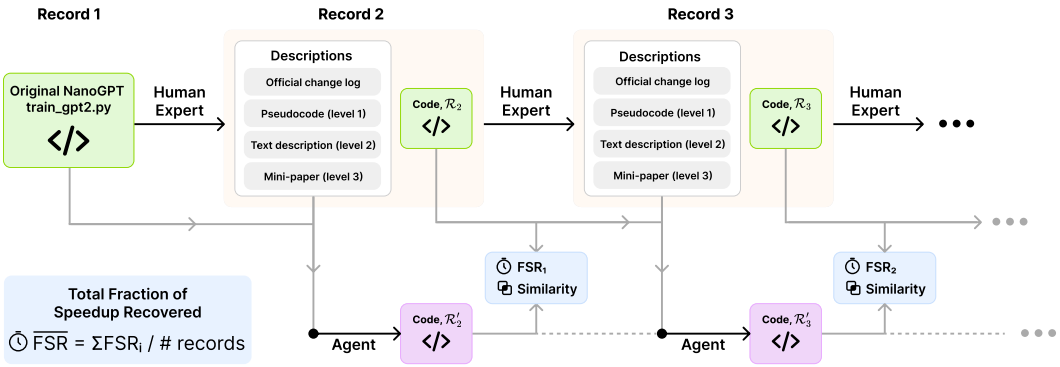

Figure 1: The Automated LLM Speedrunning Benchmark. We create a task for each consecutive pair of records $\mathcal{R}_i$, $\mathcal{R}_{i+1}$. The performance of the agent is evaluated by comparing the relative speedup of the agent solution $\mathcal{R}'_i$ to $\mathcal{R}_i$.

Abstract

Rapidly improving large language models (LLMs) have the potential to assist in scientific progress. One critical skill in this endeavor is the ability to faithfully reproduce existing work. To evaluate the capability of AI agents to reproduce complex code in an active research area, we introduce the Automated LLM Speedrunning Benchmark, leveraging the research community's contributions to the *NanoGPT speedrun*, a competition to train a GPT-2 model in the shortest time. Each of the 19 speedrun tasks provides the agent with the previous record's training script, optionally paired with one of three hint formats, ranging from pseudocode to paper-like descriptions of the new record's improvements. Records execute quickly by design and speedrun improvements encompass diverse code-level changes, ranging from high-level algorithmic advancements to hardware-aware optimizations. These features make the benchmark both accessible and realistic for the frontier problem of improving LLM training. We find that recent frontier reasoning LLMs combined with SoTA scaffolds struggle to reimplement already-known innovations in our benchmark, even when given detailed hints. Our benchmark thus provides a simple, non-saturated measure of an LLM's ability to automate scientific reproduction, a necessary (but not sufficient) skill for an autonomous research agent. Rapid advancements in large language models (LLMs) have the potential to assist in

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

scientific progress. A critical capability toward this endeavor is the ability to reproduce existing work. To evaluate the ability of AI agents to reproduce results in an active research area, we introduce the Automated LLM Speedrunning Benchmark, leveraging the research community's contributions on the *NanoGPT speedrun*, a competition to train a GPT-2 model in the shortest time. Each of the 19 speedrun tasks provides the agent with the previous record's training script, optionally paired with one of three hint formats, ranging from pseudocode to paper-like descriptions of the new record's improvements. Records execute quickly by design and speedrun improvements encompass diverse code-level changes, ranging from high-level algorithmic advancements to hardware-aware optimizations. These features make the benchmark both accessible and realistic for the frontier problem of improving LLM training. We find that recent reasoning LLMs combined with SoTA scaffolds struggle to reimplement already-known innovations in our benchmark, even when given detailed hints. Our benchmark thus provides a simple, non-saturated measure of an LLM's ability to automate scientific reproduction, a necessary (but not sufficient) skill for an autonomous research agent.

1 Introduction

The advent of LLMs capable of succeeding in challenging math, coding, and scientific reasoning domains has led to a surge of activity in applying LLM agents to the longstanding ambition of automated scientific discovery [Simon, 1995, Langley, 1987, Waltz and Buchanan, 2009, King et al., 2009, Steinruecken et al., 2019]. Early results suggest LLM-based systems can improve the productivity of human researchers, from formulating hypotheses to implementing code-based experiments to testing them [Romera-Paredes et al., 2024, Castro et al., 2025, Yin, 2025, Inizan et al., 2025].

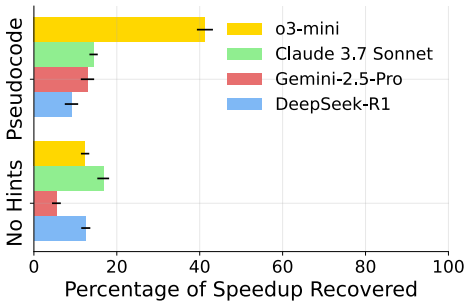

Figure 2: Recent LLM agents struggle to reproduce NanoGPT Speedrun records.

Scientific progress hinges on trustworthy results, and the ultimate test of the truth behind a finding is whether the experiment and its outcomes can be reproduced [Fineberg et al., 2019, Pineau et al., 2021, Henderson et al., 2018]. Thus, a critical component of automated science is *automated reproducibility*: the process of automatically reimplementing an experiment based on a description of the experiment design, such that the implementation reproduces previously reported outcomes. In other words, translating the description of an experiment into its implementation [Peng, 2011, Siegel et al., 2024]. Moreover, success in reimplementing a known study also serves as a metric for assessing the reliability with which an agent can implement experiments via description, an ability that would enable researchers to quickly scale up the testing of new ideas, regardless of whether they are of human or AI origin.

We study the ability of recent reasoning LLMs in combination with state-of-the-art *scaffolds*—programs that iteratively make use of an LLM for finding a solution to a given task—on reproducing prior discoveries in the domain of LLM training. We henceforth refer to the combination of a specific LLM and scaffold for the purpose of automated research as a *research agent*, and use the more specific term *AI research agent* to refer to those specifically designed for automating AI research itself. While there is much speculation that AI research agents may lead to the beginnings of a recursive self-improvement loop for future LLM-based research agents, we set our focus on the more modest goal of understanding whether current AI research agents can succeed at the prerequisite task of reproducing previous scientific findings on GPT-2 [Radford et al., 2019], the first model to demonstrate a broad capacity for zero-shot transfer to new tasks via prompting.

Towards this goal, we introduce *The Automated LLM Speedrunning Benchmark*, based on the series of community-driven improvements to GPT-2 training in the *NanoGPT Speedrun* [Jordan et al., 2024a], a competition based on minimizing the wall time of training an open-source PyTorch reimplementation of GPT-2 [Karpathy, 2023] to reach a target cross-entropy loss of 3.28 on the validation set of FineWeb [Penedo et al., 2024], using a single 8×H100 node. Since its inception in June 2024, this community effort has driven the training time of GPT-2 from 45 minutes to below 3 minutes (as of

Table 1: Key motivations of our benchmark design and how it differentiates from existing ML reproducibility benchmarks. Here, "Reproducibility" denotes whether the tasks require replicating a given technique; "Sequential", whether the benchmark measures reproducibility over a cumulative series of scientific results; "LLM research", whether the task involves language model development; and "Agent scaffold", whether a baseline agent scaffold is released with the benchmark.

	Reproducibility	**Sequential**	**LLM research**	**Agent scaffold**
MLE-bench [Chan et al., 2025]	No	No	No	No
PaperBench [Starace et al., 2025]	Yes	No	Partially	Yes
CORE-bench [Siegel et al., 2024]	Yes	No	No	Yes
RE-bench [Wijk et al., 2024]	No	No	Yes	Yes
MLAgentBench [Huang et al., 2024]	No	No	Partially	Yes
MLGym-bench [Nathani et al., 2025]	No	No	Partially	Yes
Automated LLM Speedrunning (ours)	Yes	Yes	Yes	Yes

May 2025). These improvements were driven by new algorithmic enhancements, some of which have been shown to generalize beyond the scale of the 124M parameter GPT-2 model, with the most notable being the invention of the Muon optimizer [Jordan et al., 2024b], later demonstrated to show benefits for training much larger modern LLMs [Liu et al., 2025a, Shah et al., 2025]. Other speedrun improvements include mixed precision training and more efficient attention variants [Dong et al., 2024]. As of May 2025, the NanoGPT Speedrun includes 21 successive speedrun records. Each record is associated with its corresponding training script (`train_gpt.py`), a measured training time, a public announcement of the changes, and a high-level summary of the code changes.[1]

The Automated LLM Speedrunning Benchmark then tasks an AI research agent with reproducing each successive speedrun record, starting from the previous record, with an optional set of hints of various formats and levels of detail. The clear code-level ground-truth targets per record alongside detailed change logs between records make this benchmark an ideal testing ground for the ability of agents to reproduce not only a single experimental finding, but also a series of cumulative research findings—a distinct affordance compared to prior reproducibility benchmarks. Here, all tasks share the same success metric of training time to reach the target validation loss, measured on a fixed hardware configuration (a single 8xH100 node), making exact reproduction, fair comparisons, and cross-task comparisons straightforward. Lastly, perhaps the most compelling aspect of this benchmark is its focus on reproducing discoveries directly relevant to real-world LLM development.

Our experiments show that even when given a description of the difference between two consecutive speedrun records in various formats, recent agents based on DeepSeek-R1 [DeepSeek-AI et al., 2025] and o3-mini [OpenAI, 2025] combined with a state-of-the-art search scaffold, still struggle to improve ground-truth records to match the speedups of the next ground-truth record (see Figure 2).

We believe the Automated LLM Speedrunning Benchmark can spur development of AI research agents that can automate reproducibility studies, paving a critical step on the way towards more capable AI research agents that can realize the aspiration of accelerating the pace of scientific discovery via automated science. However, our results show that before such lofty goals can be realized, automated reproducibility remains a central challenge that must be addressed.

2 Related works

Automated reproducibility. Recent works have devised benchmarks for evaluating the ability of LLM agents to reproduce code-based experiments from published papers. CORE-Bench measures an agent's ability to correctly install, execute, and interpret a paper's associated codebase and its outputs [Siegel et al., 2024]. Other benchmarks, including PaperBench [Starace et al., 2025], Papers2Code [Seo et al., 2025], AutoP2C [Lin et al., 2025], and SciReplicate [Xiang et al., 2025] test the agent's ability to convert a research paper to a codebase that replicates the reported findings or the agent's ability to formulate and test hypotheses [Chen et al., 2025, Liu et al., 2025b]. Instead of evaluating on a wide set of, often, unrelated papers as in these previous works, the Automated LLM Speedrunning Benchmark focuses on a single important overarching task of speeding up LLM training. This focus allows for a unified success metric across a diverse gradation of task complexity,

[1] https://github.com/KellerJordan/modded-nanogpt?tab=readme-ov-file#world-record-history

defined by the natural path of innovation previously discovered by human researchers. This grounding allows for not only comparison to granular, ground-truth code-level changes, but also opens the door to evaluating an LLM agent's ability to reproduce an entire research arc over multiple compounding innovations against human performance. Moreover, the benchmark's multiple hint levels allow for controlled study of how performance varies across different forms of background information.

Code generation with LLMs. Code is inherently reproducible via repeated execution and requires no additional equipment to run beyond a computer. Thus, many automated scientific reproducibility benchmarks, including ours, focus primarily on virtual, code-based experiments. In this domain, research agents directly benefit from and build upon the rapid progress in coding and computer-use agents, such as a growing set of complex, sandboxed software-engineering agent benchmarks [Yang et al., 2024, Wang et al., 2024, Fourney et al., 2024, Mialon et al., 2023, Yoran et al., 2024, Zhou et al., 2023, Koh et al., 2024] and scaffold designs [Zhang et al., 2024], such as AIDE [Jiang et al., 2025], which we both use as a baseline and extend in our experiments.

LLMs for automated ML. Recent advances enabling LLMs to exploit chain-of-thought outputs during inference have led to drastic improvements in their performance on reasoning tasks in domains like math, coding, and science. These improvements have led to a surge in LLM programs seeking to automate the key parts of machine learning itself, encompassing iterated hypothesis generation and testing and the writing of reports detailing the findings, in the form of end-to-end agents [Lu et al., 2024, Huang et al., 2025, Yamada et al., 2025a], agents focused on hypothesis generation [Gottweis et al., 2025, O'Neill et al., 2025], as well as agents that can interact with a human-in-the-loop to jointly formulate and test hypotheses [Intology AI, 2025, Autoscience Institute, 2025]. However, early results suggest these systems, while capable of optimizing code-level improvements, often fall short in executing on experiments that faithfully reflect their intended goals [Yamada et al., 2025b]. Thus, while LLM-based reasoning models can generate, at times, novel hypotheses [Gu et al., 2024], their ability for scientific reproduction remains a crucial bottleneck in automating scientific research.

3 The Automated LLM Speedrunning Benchmark

The Automated LLM Speedrunning Benchmark seeks to evaluate an LLM agent's ability to reproduce the wall-time speedup associated with each record transition from the NanoGPT Speedrun, both with and without access to hints describing the corresponding changes at varying levels of abstraction. Table 1 summarizes how our work compares to existing ML reproducibility benchmarks.

3.1 Reproducibility tasks from existing records

For each transition from record $\mathcal{R}_{i-1}$ to record $\mathcal{R}_i$ for $i = 2, ..., 21$, excluding $i = 7$, whose speedup is purely due to upgrading PyTorch, we define the following components:

- $\mathcal{R}_i$ Training script for the i-th record in the speedrun,
- t_i Wall-clock time (in seconds) required by $\mathcal{R}_i$ to reach the target validation loss,
- Δ_i^1 Level 1 hint: A *pseudocode* description of code change from the previous record,
- Δ_i^2 Level 2 hint: A *natural-language* description of the code change from the previous record,
- Δ_i^3 Level 3 hint: A *mini-paper* summarizing the code change from the previous record.

All hints were first drafted by R1, manually verified, and, where necessary, edited for correctness and relevance. See Appendix F for further details on our hint creation process. We provide a categorized listing of all ground-truth records in Appendix G and example hints in Appendix H.

For convenience, we denote the set of ground-truth speedrun records (which excludes record 6) as $\mathcal{I}$. We define a *record task* as a tuple $\langle \mathcal{R}_{i-1}, \mathcal{R}_i, t_i, m \rangle$, where $\mathcal{R}_1$ corresponds to the initial NanoGPT training script, and where m is any subset of the set of *hint levels*, $\{0, 1, 2, 3\}$, where level 0 corresponds to *no hint*. Depending on the presence of hints, we categorize the possible tasks in our benchmark into two types:

Record reproduction tasks. Given hints that describe the subsequent record, i.e. $m \neq \{0\}$, the LLM agent must reproduce record $\mathcal{R}_{i+1}$ given $\mathcal{R}_i$ and the set of corresponding hints. Here the key metric of interest is the *fraction of speedup recovered* (FSR), defined as

$$\text{FSR}_i = \frac{t_i - t'_{i+1}}{t_i - t_{i+1}}. \tag{1}$$

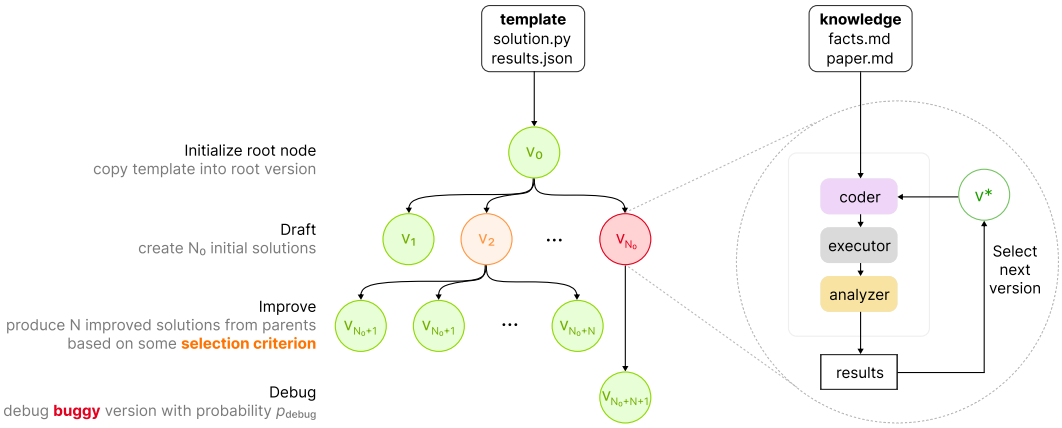

Figure 3: Overview of our flexible search scaffold. Search starts from a root node containing code for the starting record $\mathcal{R}_i$ from which N_0 initial solutions are generated. Subsequently, each search iteration debugs a buggy leaf node with probability p_{debug} and otherwise greedily selects the best node to improve, with debug and improvement each branching N solutions. At each search step, the coder submodule implements the solution, with optional access to external knowledge (e.g. hints).

where t'_{i+1} is the training time achieved by the agent to reach the target validation loss. The full benchmark performance is then the mean FSR over the set of all included records, $\mathcal{I}$:

$$\overline{\text{FSR}} = \frac{1}{|\mathcal{I}|} \sum_i \frac{t_i - t'_{i+1}}{t_i - t_{i+1}}. \tag{2}$$

Record optimization tasks. Without any hints, i.e. $m = \{0\}$, the LLM agent must produce a new training script solution $\mathcal{R}'_{i+1}$ with a minimal training time t'_{i+1} to reach the target validation loss, given $\mathcal{R}_i$. Here we consider both the raw wall time t'_{i+1} of the solution produced, in addition to FSR_i. Similar to the setting of record reproduction, we consider the mean of these metrics over all ground-truth records in the benchmark as an overall measure of performance. This allows the agent to explore its own improvements given the same SoTA starting point that humans had when each record was produced.

3.2 Agent scaffolds

We provide a flexible search scaffold implementation that extends AIDE [Jiang et al., 2025] into a more general parameterization. In this setup, visualized in Figure 3, each node in the search tree represents a solution instance contained in a directory with relevant scripts, performance metrics, and an LLM-generated execution summary. For instance, in NanoGPT training, a solution node consists of a single `train_gpt2.py` script and a results file describing its performance and execution outcome. The fitness of each node is evaluated based on these metrics—such as wall time to reach the target validation loss—with each new search initialized using a ground-truth script from the benchmark and proceeding by branching into up-to-multiple child solutions.

Each search step follows three stages: implementation, execution, and analysis. During implementation, the agent generates working code from a prompt that includes the task description and optionally, a set of associated hints. We use Aider [Gauthier, 2025] to make diff-based edits to the initial solution, producing modified versions for execution. These solutions are then run on an 8xH100 node, and the output is summarized in natural language via the analysis stage, capturing key performance indicators and insights from standard outputs. Custom prompts guide each stage and are detailed in Appendix F. The search begins with N_0 initial modifications to the root node. At each step, a new node branches from either a randomly chosen buggy node (with probability p_{debug}) or the highest-performing node. To avoid redundant debugging, we cap retries at D_{max} per node. This scaffold design supports multiple search variants, outlined in Table 2, with each receiving the same budget M of search steps to ensure fair comparison.

Table 2: Search variants and their corresponding scaffold parameterizations.

Method	Initial branch factor	Branch factor	Debug probability	Max debug depth
Tree	1	N	0	0
Forest	N_0	N	0	0
AIDE	N_0	1	p_{debug}	D_{max}
Multi-AIDE	N_0	N	p_{debug}	D_{max}
Flat (Best-of-M)	M	—	—	—

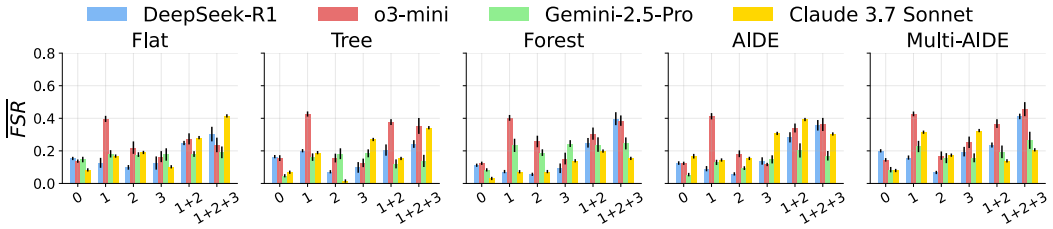

Figure 4: Mean FSR across five search variants and four frontier models for six hint regimes: no hint (0), pseudocode (1), text (2), mini-paper (3) and combinations thereof ($1+2$, $1+2+3$).

4 Experiments and results

We now evaluate the performance of several baseline agents across a range of scaffolds, hint formats, and model backbones for all NanoGPT Speedrun records. We report results using the normalized runtime improvement metric (FSR) from Equation 2, as well as measures of code similarity between agent and human solutions. For fair comparisons, we use training times for human records based on rerunning each ground-truth record on the same hardware configuration as agent solutions. Appendix C reports the near exact reproduction of training times for human records on our cluster.

4.1 Baselines

We compare a number of LLM agents based on DeepSeek-R1, o3-mini, Gemini-2.5-Pro, and Claude-3.7-Sonnet, using instances of the search scaffolds listed in Table 2. Our choice of parameters are $N_0 = 3$ for the initial pool of root hypotheses (forest, AIDE and multi-AIDE), $N = 3$ for the branching factor (tree, forest and multi-AIDE), $p_{\text{debug}} = 0.5$ and $D_{\text{max}} = 5$ for the debug probability and maximum debug depth respectively (AIDE and multi-AIDE), and a search budget of $M = 20$ nodes. Taken together, these scaffolds cover a range of branching factors, search depth, and debug logic.

For each pair of model and search scaffold, we assess the mean FSR across all 19 tasks for each of the following hint levels: no hint (level 0), pseudocode (level 1), text description (level 2), and mini-paper (level 3). Each solution is executed under a maximum runtime of 60 minutes (i.e. a maximum of 20 hours per agent run). We observe an average run time of $\approx$10 hours per agent run, across a total of 6,840 agent runs (19 records $\times$ 6 hint regimes $\times$ 5 search variants $\times$ 4 models $\times$ 3 seeds), for a total of 6,840 $\times$ 8 H100 (internal cluster) hours spent executing the generated solutions.

4.2 Reproducing individual records

We report the mean FSR for each model, search scaffold, and hint-level combination across 3 full search runs in Figure 4, including the case of no hints. It is evident that hints are necessary for inducing greater values of FSR, with all agents failing to recover more than 20% of the speed-up achieved by human solutions on average without hints. Appendix D further reports the mean FSR for each individual record transitions per agent variation across 3 runs per variation.

We observe that o3-mini generally achieves equal or better results than other models in mean FSR for all hint levels, but sees slightly worse performance with no hints. Notably, flat search (i.e. best-of-M), generally matches or outperforms iterated search scaffolds across the individual hint levels (levels 1–3), while matching their performance in the case of no hints. Moreover, tree and forest methods, which lack debug steps, perform on par with AIDE-based search scaffolds, suggesting that explicit debug steps do not provide a significant benefit on top of iterative improvement steps. Overall, the

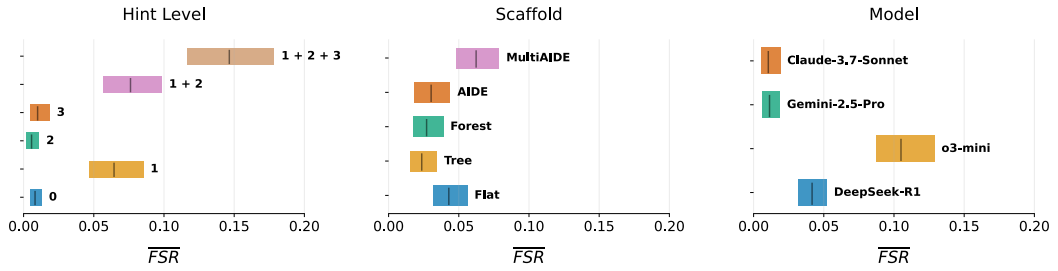

Figure 5: Interquartile Mean (IQM) evaluation results. Scores are aggregated across multiple runs with the same hint level, scaffold, and model.

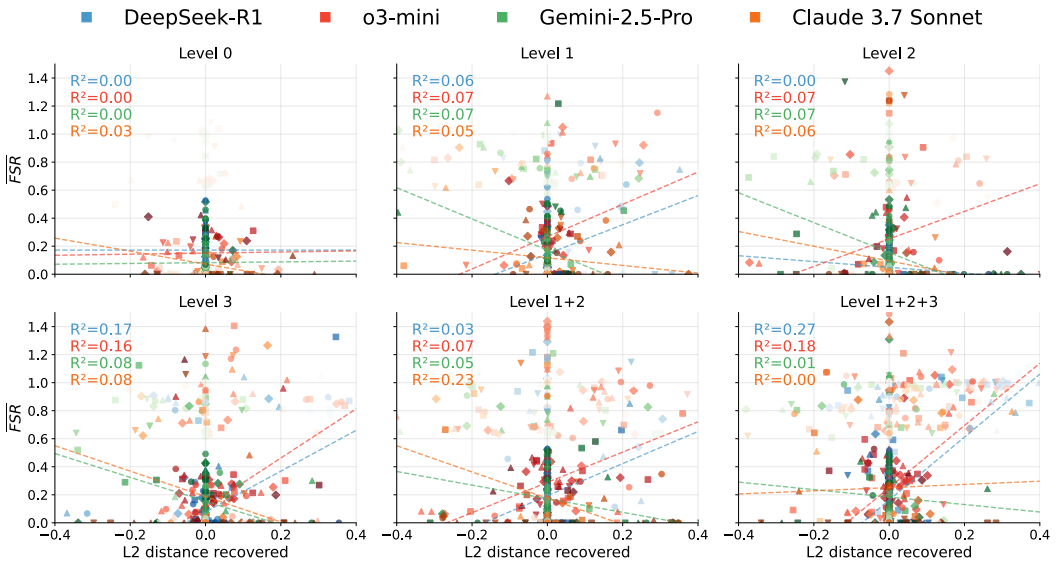

Figure 7: Correlation of FSR with L2 distance recovered for each hint level, showing a modest correlation between similarity to the human solution and FSR for most hint types and models.

gap between the best models (o3-mini and Claude-3.7-Sonnet) and the open-weights (R1) is wider for the search scaffolds incorporating branching logic (tree, search, and AIDE variants), suggesting that models like o3-mini can better iterate on their previous solutions. Figure 6 further shows how agents tend to have more difficulty in reproducing later records.

Out of the various hint formats, the most useful are the pseudocode and the combinations of pseudocode with text and mini-papers hints, which enable o3-mini to recover approximately 40% and 46%, respectively, of the speed-up attained by human solutions on average. Surprisingly, R1 agents seem to worsen with the presence of the individual hints, generally achieving lower FSR compared to the no-hint setting, suggesting that attempting to implement the complex changes in these hints results in buggy code. With hints, R1 produces solutions with lower FSR than simply making no changes to the code, a common outcome with no hints, as indicated by the cluster around a recovered L2 embedding distance of 0.0 in Figure 7 (Section 4.6 details this similarity analysis).

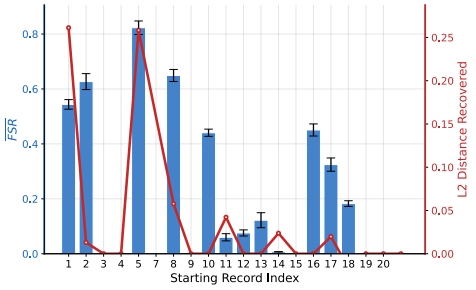

Figure 6: FSR and embedding distance per record for o3-mini with text description hints (mean and std over 3 seeds). Later records tend to be harder for agents, leading to lower recovered embedding distance and speedups.

4.3 Combining multiple hints

We further investigate the impact of combining hint formats, and also include these results for each agent variation in Figure 4. We observe that providing the text description or mini-paper together

Table 3: Performance comparison across different hint formats (mean and std over 3 runs). Color-coded values are differences relative to the best-performing individual hint in the combination.

Hints	Model	Flat	Tree	Forest	AIDE	Multi-AIDE
L1 (pseudocode)	o3-mini	**0.40**±0.02	**0.43**±0.02	**0.40**±0.02	**0.41**±0.02	0.43±0.02
L2 (text)	o3-mini	0.22±0.04	0.16±0.03	0.26±0.04	0.18±0.02	0.17±0.03
L3 (mini-paper)	o3-mini	0.17±0.03	0.13±0.03	0.15±0.04	0.12±0.01	0.25±0.04
L1+L2	o3-mini	0.27±0.03 (-0.13)	0.38±0.02 (-0.05)	0.31±0.04 (-0.09)	0.34±0.03 (-0.07)	0.37±0.03 (-0.06)
L1+L2+L3	o3-mini	0.24±0.05 (-0.16)	0.35±0.05 (-0.08)	0.39±0.03 (-0.01)	0.36±0.04 (-0.05)	**0.46**±0.04 (+0.03)
L1 (pseudocode)	DeepSeek-R1	0.13±0.03	0.20±0.00	0.07±0.00	0.09±0.02	0.16±0.01
L2 (text)	DeepSeek-R1	0.10±0.01	0.07±0.00	0.06±0.00	0.06±0.01	0.07±0.00
L3 (mini-paper)	DeepSeek-R1	0.13±0.04	0.10±0.03	0.09±0.03	0.14±0.02	0.20±0.03
L1+L2	DeepSeek-R1	0.25±0.01 (+0.12)	0.20±0.03 (+0.00)	0.25±0.03 (+0.18)	0.28±0.03 (+0.19)	0.24±0.02 (+0.08)
L1+L2+L3	DeepSeek-R1	**0.30**±0.04 (+0.17)	**0.24**±0.02 (+0.04)	**0.40**±0.04 (+0.31)	**0.36**±0.03 (+0.22)	**0.41**±0.02 (+0.21)
L1 (pseudocode)	Gemini-2.5-Pro	0.18±0.02	0.16±0.02	0.23±0.04	0.13±0.02	0.23±0.03
L2 (text)	Gemini-2.5-Pro	0.18±0.01	0.18±0.03	0.19±0.02	0.09±0.01	0.16±0.03
L3 (mini-paper)	Gemini-2.5-Pro	0.18±0.04	**0.18**±0.02	0.24±0.02	0.15±0.02	0.16±0.03
L1+L2	Gemini-2.5-Pro	0.18±0.02 (+0.00)	0.12±0.03 (-0.06)	0.24±0.04 (+0.01)	**0.20**±0.04 (+0.07)	0.19±0.04 (-0.04)
L1+L2+L3	Gemini-2.5-Pro	**0.19**±0.04 (+0.01)	0.14±0.04 (-0.04)	**0.25**±0.04 (+0.01)	0.17±0.03 (+0.02)	**0.26**±0.05 (+0.03)
L1 (pseudocode)	Claude-3.7-Sonnet	0.14±0.03	0.13±0.03	0.05±0.01	0.14±0.01	0.18±0.04
L2 (text)	Claude-3.7-Sonnet	0.10±0.03	0.03±0.01	0.06±0.02	0.14±0.02	0.14±0.02
L3 (mini-paper)	Claude-3.7-Sonnet	0.06±0.02	0.22±0.02	0.11±0.01	**0.34**±0.01	0.19±0.03
L1+L2	Claude-3.7-Sonnet	0.14±0.03 (+0.00)	0.11±0.02 (-0.11)	**0.15**±0.02 (+0.04)	0.30±0.02 (-0.04)	0.09±0.01 (-0.09)
L1+L2+L3	Claude-3.7-Sonnet	**0.21**±0.04 (+0.07)	**0.31**±0.02 (+0.09)	0.10±0.02 (-0.01)	0.31±0.01 (-0.03)	**0.20**±0.02 (+0.01)

with the pseudocode compared to only providing the pseudocode hint can substantially degrade performance for o3-mini (see o3-mini result in Table 3), but surprisingly benefits R1. These results suggest that o3-mini may be less capable of taking advantage of longer contexts, while R1's reasoning directly benefits from longer initial prompts. On the other hand, the effect of combined hints on Gemini-2.5-Pro and Claude-3.7-Sonnet appears relatively small, suggesting they can handle longer context yet lacks the ability to leverage them for effective reasoning for reproducing code changes.

4.4 Interquartile mean evaluation

As the agent runs could bring a large variance in the experimental results, in Figure 5 we present the aggregated Interquartile Mean (IQM) results across runs with the same hint level, search scaffold, and model. The IQM metric has been shown to be robust to comparisons with a small sample size, and in Figure 5 we report as 95% confidence intervals, bootstrapped from 3 seeds following Agarwal et al. [2021]. On the hint level comparison, the agents reach the best performance when using all three hints combined. For individual hints, the pseudo-code hint performs the best. For search scaffold, multi-AIDE search outperforms all others. On the model side, we are surprised to find that the Gemini-2.5-Pro and Claude-3.7-Sonnet gives the lowest performance close to 0 FSR, even lagging behind the open-sourced R1 model. The results also suggest Automated LLM Speedrunning is a challenging benchmark for current agents as the aggregated performances are fairly low.

4.5 Analysis of search trees

To better understand how each agent spends its search budget, we inspect the proportion of different kinds of nodes in their search trees: buggy nodes, which crash due to runtime errors; improved nodes, which successfully improved runtime compared to their parent; and unimproved nodes, which do not improve from their parent. This breakdown of the search trees is presented in Figure 8. We observe that flat search leads to a higher total proportion of buggy nodes, indicating that initially-proposed solutions are most often incorrect. We also notice that R1 agents generate more buggy nodes under AIDE and multi-AIDE—the two variants with debugging steps—suggesting that R1 may be less capable of fixing its own mistakes compared to o3-mini. Gemini-2.5-Pro tends to generate fewer buggy nodes compared to the other models, yet it lags behind on the FSR metric (see Figure 4 and Figure 2), suggesting that Gemini produces more robust code at the cost of correctly implementing the more efficient solutions described in the hints. Surprisingly, Claude-3.7-Sonnet generates significantly more buggy nodes than the other three models, with the fraction of buggy nodes gradually overtaking the fraction of working nodes in the search tree, indicating that Claude-3.7-Sonnet struggles to improve and debug its previous solutions.

The analysis of node types in the search tree provides insight into the discrepancy on the results of Claude-3.7-Sonnet between the $\overline{FSR}$ results in Figure 4 and the IQM results in Figure 5. While the

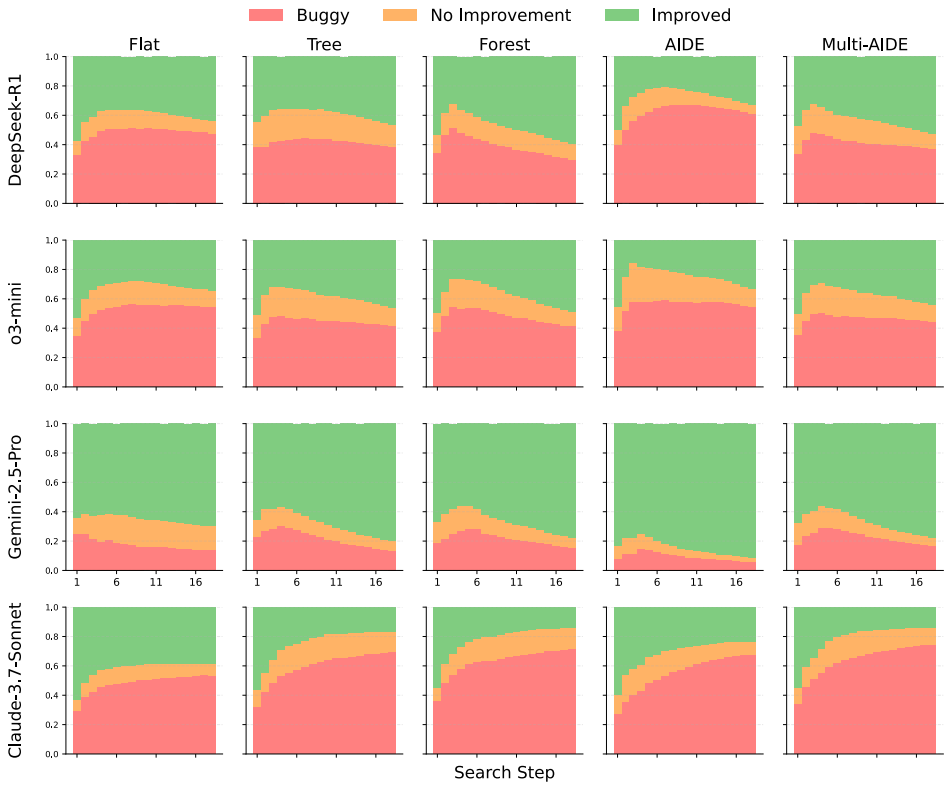

Figure 8: Fraction of node types across search trees for each model and search method. Notably, branching (i.e. non-flat) search is beneficial for reducing the proportion of buggy nodes. Further, a majority of non-buggy steps produce improved nodes for all branching search methods.

$\overline{FSR}$ results suggest that, on average, Claude-3.7-Sonnet performs comparably to o3-mini, the IQM plot indicates that Claude-3.7-Sonnet significantly lags behind o3-mini. This discrepancy can be explained by examining the distribution of node types in the search tree. Claude-3.7-Sonnet is capable of generating working solutions that substantially improve the FSR. However, it also produces a considerable number of buggy nodes that result in runtime errors. These errors negatively impact the overall performance as reflected in the IQM plot, despite the improvements in the averaged $\overline{FSR}$.

4.6 Similarity between agent and human solutions

Agents may output solutions with similar performance to human ones, but may still fail to reproduce the target code changes. We thus assess code similarity between agent and human solutions by comparing code embedding distances using the SFR-Embedding-Code 2B model [Liu et al., 2024].

Specifically, we normalize the embedding distance between the agent's code solution and the target human solution, i.e. the next record, and divide this distance by the embedding distance between the current and the next human record. Figure 7 depicts the normalized L2 embedding distance recovered with respect to the record speedups and for each type of hint. Here the distance recovered is defined as $1 - \|e_{i+1} - e'_{i+1}\|/\|e_{i+1} - e_i\|$, where e_i is the embedding for $\mathcal{R}_i$, and e'_i is the embedding for the LLM's attempt at reproducing it. We observe a stronger correlation between higher similarity score and FSR for richer hint formats, suggesting that distances under this embedding space can be a meaningful measure of degree of successful reproduction.

As an alternative measure of code similarity, we made use of R1 as a judge, prompting it to assess what fraction of the ground-truth code changes between the current and next record were successfully reproduced in the agent's solution, on a scale of 0 to 1 with a score of 1 corresponding to a completely correct reimplementation. Appendix E contains a comparison between these judge-based similarity scores and FSR across all agent variations. We observe clear positive correlation between higher similarity scores and FSR. We provide sample outputs from R1 judge in Appendix F.

5 Limitations and future directions

Our Automated LLM Speedrunning Benchmark serves as a challenging evaluation of an LLM agent's ability to reproduce scientific findings specific to LLM training. However, there remain important limits in its capacity for assessing an agent's true capability for scientific reproduction, and each of these limitations point the way to directions for exciting future research.

Scaling up external knowledge. By design, the various hint levels are succinct and easily fit within the context of the LLMs we tested. Moreover, these hints were manually defined, with the relevant hint directly provided as part of the associated task instance. A more realistic setup would provide the agent with the ability to use external knowledge via some form of function calling, including the ability to store intermediate results in various kinds of memory structures, e.g. a short-term scratchpad, long-term database, or neural module [Hermann et al., 2015, Weston et al., 2014]. Accessing a wider and potentially accumulating set of external information would also test the agent's ability to manage information whose total size may exceed its context length [Sarthi et al., 2024].

Memorization or generalization? As many of the ground-truth records in the NanoGPT Speedrun were published potentially before the cut-off date of the models used in our experiments (and thus, most likely of future models), there is the possibility that models may have already seen these solutions during training [Gupta and Pruthi, 2025]. We find that neither R1 nor o3-mini accurately reproduce the speedups realized in the ground-truth records, but explicitly disentangling memorization from generalization may become more necessary as models begin to saturate the benchmark. More advanced techniques for measuring memorization in LLMs would allow for a more nuanced evaluation [Carlini et al., 2021, Razeghi et al., 2022, Oren et al., 2023, Deng et al., 2024].

Semantic diffs. Our experiment analysis focuses on FSR and numeric similarity scores between the LLM's solution and the corresponding human solution. Moving beyond a similarity score toward more expressive natural-language summaries, e.g. via automatically-generated commit messages [Jiang et al., 2017], of the code diffs between LLM and human solutions would allow for more scalable identification of common mistakes or new innovations with respect to the human solutions.

From LLM speedrun to ML speedrun. The skills required for the LLM speedrun are a good starting point but are not enough to create reliable AI research agents. True research agents must handle more complex tasks, such as working with entire multi-file codebases, optimizing for metrics beyond training time like model performance or memory usage, dealing with distributed training, and defining their own success metrics. Most importantly, the current benchmark tests the ability to reproduce results, not to innovate. While an LLM beating human records would be a milestone, the ultimate test is whether future agents can solve new, open scientific challenges.

6 Conclusions

We introduced the Automated LLM Speedrunning Benchmark, a challenging evaluation of an LLM agent's ability to reproduce existing scientific innovations in LLM training, based on reproducing each successive record in the community-driven NanoGPT Speedrun. Unlike previous benchmarks for automated scientific reproducibility, our benchmark enables evaluations of an agent's ability to reproduce not just a single result, but each incremental advance across a chain of research innovations. We found that even recent, leading reasoning models, like R1 and o3-mini, when combined with a state-of-the-art agent scaffold, still struggle to successfully produce speedrun solutions that match the speedups attained by the corresponding human solutions. Moreover, this gap between human and agent performance persists even when these strong baseline agents are provided with detailed explanations describing the exact code changes from the previous speedrun record. Our results suggest that automated reproducibility may serve as a significant obstacle in realizing reliable, autonomous research agents with current, leading models, and we expand on the potential societal impacts of our work in Appendix I. We believe the Automated LLM Speedrunning Benchmark can be an effective testbed for monitoring this crucial capability in future research agents.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Autoscience Institute. Carl technical report, 2025. URL https://www.autoscience.ai/blog/meet-carl-the-first-ai-system-to-produce-academically-peer-reviewed-research.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650, 2021.

Pablo Samuel Castro, Nenad Tomasev, Ankit Anand, Navodita Sharma, Rishika Mohanta, Aparna Dev, Kuba Perlin, Siddhant Jain, Kyle Levin, Noémi Éltető, et al. Discovering symbolic cognitive models from human and animal behavior. *bioRxiv*, pages 2025–02, 2025.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. Mle-bench: Evaluating machine learning agents on machine learning engineering, 2025. URL https://arxiv.org/abs/2410.07095.

Tingting Chen, Srinivas Anumasa, Beibei Lin, Vedant Shah, Anirudh Goyal, and Dianbo Liu. Auto-bench: An automated benchmark for scientific discovery in llms. *arXiv preprint arXiv:2502.15224*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models, 2024. URL https://arxiv.org/abs/2311.09783.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels, 2024. URL https://arxiv.org/abs/2412.05496.

Harvey Fineberg, National Academies of Sciences, and Medicine. *Reproducibility and replicability in science*. National Academies Press, 2019.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang, Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-one: A generalist multi-agent system for solving complex tasks, 2024. URL https://arxiv.org/abs/2411.04468.

Paul Gauthier. Aider: Ai pair programming in your terminal. https://github.com/Aider-AI/aider, 2025. URL https://github.com/Aider-AI/aider. Version 0.82.0.

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

Tianyang Gu, Jingjin Wang, Zhihao Zhang, and HaoHong Li. Llms can realize combinatorial creativity: generating creative ideas via llms for scientific research. *arXiv preprint arXiv:2412.14141*, 2024.

Tarun Gupta and Danish Pruthi. All that glitters is not novel: Plagiarism in ai generated research, 2025. URL https://arxiv.org/abs/2502.16487.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Karl Moritz Hermann, Laurent Orseau, Shaodi Wang, Soham Tunyasuvut, Hubert Stanczyk, and Charles Blundell. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Kexin Huang, Ying Jin, Ryan Li, Michael Y Li, Emmanuel Candès, and Jure Leskovec. Automated hypothesis validation with agentic sequential falsifications. *arXiv preprint arXiv:2502.09858*, 2025.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation, April 2024. URL https://arxiv.org/abs/2310.03302.

Theo Jaffrelot Inizan, Sherry Yang, Aaron Kaplan, Yen-hsu Lin, Jian Yin, Saber Mirzaei, Mona Abdelgaid, Ali H Alawadhi, KwangHwan Cho, Zhiling Zheng, et al. System of agentic ai for the discovery of metal-organic frameworks. *arXiv preprint arXiv:2504.14110*, 2025.

Intology AI. Zochi technical report, 2025. URL https://github.com/IntologyAI/Zochi/blob/main/Zochi_Technical_Report.pdf.

Siyuan Jiang, Ameer Armaly, and Collin McMillan. Automatically generating commit messages from diffs using neural machine translation. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 135–146. IEEE, 2017.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL https://github.com/KellerJordan/modded-nanogpt.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024b. URL https://kellerjordan.github.io/posts/muon/.

Andrej Karpathy. nanogpt, 2023. URL https://github.com/karpathy/nanoGPT.

Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923): 85–89, 2009.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

Pat Langley. *Scientific discovery: Computational explorations of the creative processes*. MIT press, 1987.

Zijie Lin, Yiqing Shen, Qilin Cai, He Sun, Jinrui Zhou, and Mingjun Xiao. Autop2c: An llm-based agent framework for code repository generation from multimodal content in academic papers, 2025. URL https://arxiv.org/abs/2504.20115.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a.

Ye Liu, Rui Meng, Shafiq Joty, Silvio Savarese, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Codexembed: A generalist embedding model family for multiligual and multi-task code retrieval, 2024. URL https://arxiv.org/abs/2411.12644.

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

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. Mlgym: A new framework and benchmark for advancing ai research agents, 2025. URL https://arxiv.org/abs/2502.14499.

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.

OpenAI. Openai o3-mini: Pushing the frontier of cost-effective reasoning, January 2025. URL https://openai.com/index/openai-o3-mini. Accessed: 2025-05-14.

Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B. Hashimoto. Proving test set contamination in black box language models, 2023. URL https://arxiv.org/abs/2310.17623.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Roger D Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of machine learning research*, 22(164):1–20, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*, 2022.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024.

Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning, 2025. URL https://arxiv.org/abs/2504.17192.

Ishaan Shah, Anthony M Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

Zachary S Siegel, Sayash Kapoor, Nitya Nagdir, Benedikt Stroebl, and Arvind Narayanan. Core-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *arXiv preprint arXiv:2409.11363*, 2024.

Herbert Simon. Machine discovery. *Foundations of Science*, 1:171–200, 1995.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. Paperbench: Evaluating ai's ability to replicate ai research, 2025. URL https://arxiv.org/abs/2504.01848.

Christian Steinruecken, Emma Smith, David Janz, James Lloyd, and Zoubin Ghahramani. The automatic statistician. *Automated machine learning: Methods, systems, challenges*, pages 161–173, 2019.

David Waltz and Bruce G Buchanan. Automating science. *Science*, 324(5923):43–44, 2009.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents, 2024. URL https://arxiv.org/abs/2407.16741.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, et al. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. *arXiv preprint arXiv:2411.15114*, 2024.

Yanzheng Xiang, Hanqi Yan, Shuyin Ouyang, Lin Gui, and Yulan He. Scireplicate-bench: Benchmarking llms in agent-driven algorithmic reproduction from research papers. *arXiv preprint arXiv:2504.00255*, 2025.

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*, 2025a.

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, 2025b. URL https://arxiv.org/abs/2504.08066.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, May 2024. URL https://arxiv.org/abs/2405.15793.

Weiguo Yin. Exact solution of the frustrated potts model with next-nearest-neighbor interactions in one dimension: An ai-aided discovery. *arXiv preprint arXiv:2503.23758*, 2025.

Ori Yoran, Samuel Joseph Amouyal, Chaitanya Malaviya, Ben Bogin, Ofir Press, and Jonathan Berant. Assistantbench: Can web agents solve realistic and time-consuming tasks? *arXiv preprint arXiv:2407.15711*, 2024.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 1592–1604, 2024.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We include in the main body of the paper the contributions described in the abstract and introduction: benchmark construction (Section 3), baselines varying frontier model, underlying scaffold and hint format (Section 4.2) and a similarity analysis (Section 4.6).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 contains a dedicated discussion on limitations of our work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide a detailed description of our methodology and our experimental runs, including important hyperparameter values and prompts used (in Section 4 and Appendices F)–H. We are accompanying our submission with a zip file containing scripts for running our experiments and notebooks for carrying out the data analyses.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We are including the benchmark data, the agent implementation we are using, the scripts to run the experiments and the notebooks for analyses in a zip file accompanying our submission. This content is also linked in the main body of the paper in Section 1.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide a detailed overview of our agentic scaffold methodology, the frontier models utilized and the various hint formats used for each experiment in the main body of the paper (see Sections 3 – 4). The exact implementation details, including prompts used, can be recovered by the code contained in the zip file accompanying the submission.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer:[Yes]

   Justification: We report error bars for our set of baselines and additional experiments, which are based on standard deviation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide a description of the hardware we used and an estimate of the amount of the resources needed to run the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer:[Yes]

   Justification: We have reviewed the linked NeurIPS Code of Ethics and can confirm the the research conducted in the paper conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: The benchmark release described in the paper aims to drive progress in foundational research by improving the ML engineering capabilities of LLM agents. This can have long-term societal impacts which we briefly touch upon in the introduction and extensively discuss in Appendix I.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the original resource of the assets we are using, we provide a URL and we explicitly name their license.

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We make all our code and datasets available in an anonymized .zip file uploaded in an anonymous URL. We accompany our submission with a Croissant metadata file to document our dataset.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We have used open-source LLMs for generating data that form part of the benchmark (specifically, the various pseudocode, text and mini-paper hints). Querying LLMs is also a core part of the various stages comprising the improvement step of the LLM agent (implementation, execution and analysis). Finally, we are leveraging an open-source LLM to obtain LLM judge scores and build a similarity metric that estimates proximity of the agent-generated code with the human code. We are documenting these usages throughout the paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.