FAIR

The Automated LLM Speedrunning Benchmark: Reproducing NanoGPT Improvements

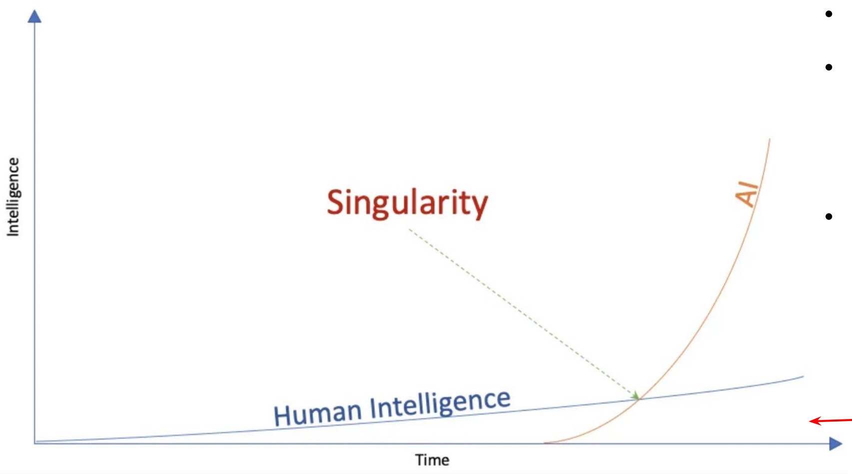
NeurIPS 2025

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]

[1]Meta, [2]University of Edinburgh
[*]Equal contribution

Introduction / Motivation



🧪 Towards automated scientific discovery

- LLMs have been becoming increasingly capable in math, coding and scientific reasoning domains.
- We already see instances where LLM-based systems can improve the productivity of human researchers.
- And many groups are working on end-to-end AI research agents (encompassing iterated hypothesis generation, testing, and writing reports detailing findings).

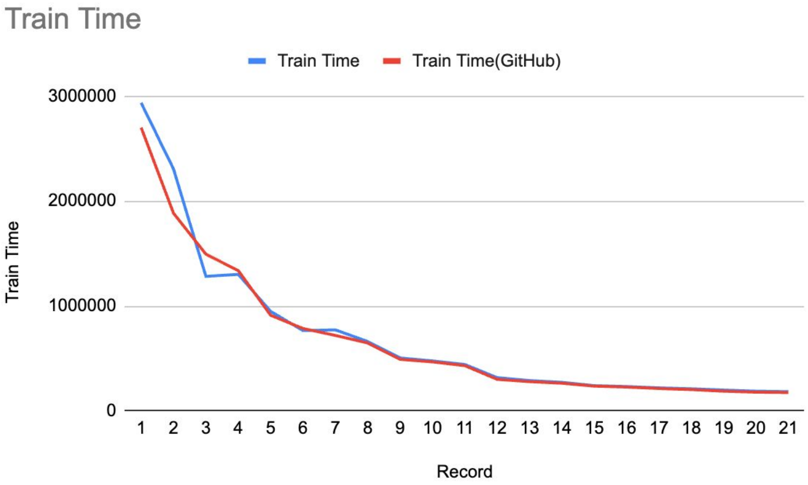
✅ Reproducibility is a critical component of science

- Scientific progress hinges on trustworthy results.
- Automated science needs automated reproducibility (reimplementing an experiment based on a description of the experiment design and reproducing reported results).
- Necessary (but not sufficient) skill for automated scientific discovery.

Where are we in this graph?

NanoGPT Speedrun

#	Record time	Description	Date	Log	Contributors
1	45 minutes	llm.c baseline	05/28/24	log	@karpathy, llm.c contributors
2	31.4 minutes	Tuned learning rate & rotary embeddings	06/06/24	log	@kellerjordan0
3	24.9 minutes	Introduced the Muon optimizer	10/04/24	none	@kellerjordan0, @jxbz
4	22.3 minutes	Muon improvements	10/11/24	log	@kellerjordan0, @bozavlado
5	15.2 minutes	Pad embeddings, ReLU², zero-init projections, QK-norm	10/14/24	log	@Grad62304977, @kellerjordan0
6	13.1 minutes	Distributed the overhead of Muon	10/18/24	log	@kellerjordan0



- Community-driven improvements to GPT-2 training. Competition based on minimizing wall time of training a GPT-2 implementation to reach a target cross-entropy loss of 3.28 on validation set of FineWeb.
- (As of May 2025) 21 records reducing training time from 45 → 3 min. Encompasses diverse code-level changes, ranging from high-level algorithmic to hardware-aware optimizations.
- Can AI research agents reproduce each record with sets of hints of various formats and levels of detail?
- LLM pre-training task which can help monitoring for recursive self improvement abilities.

Benchmark

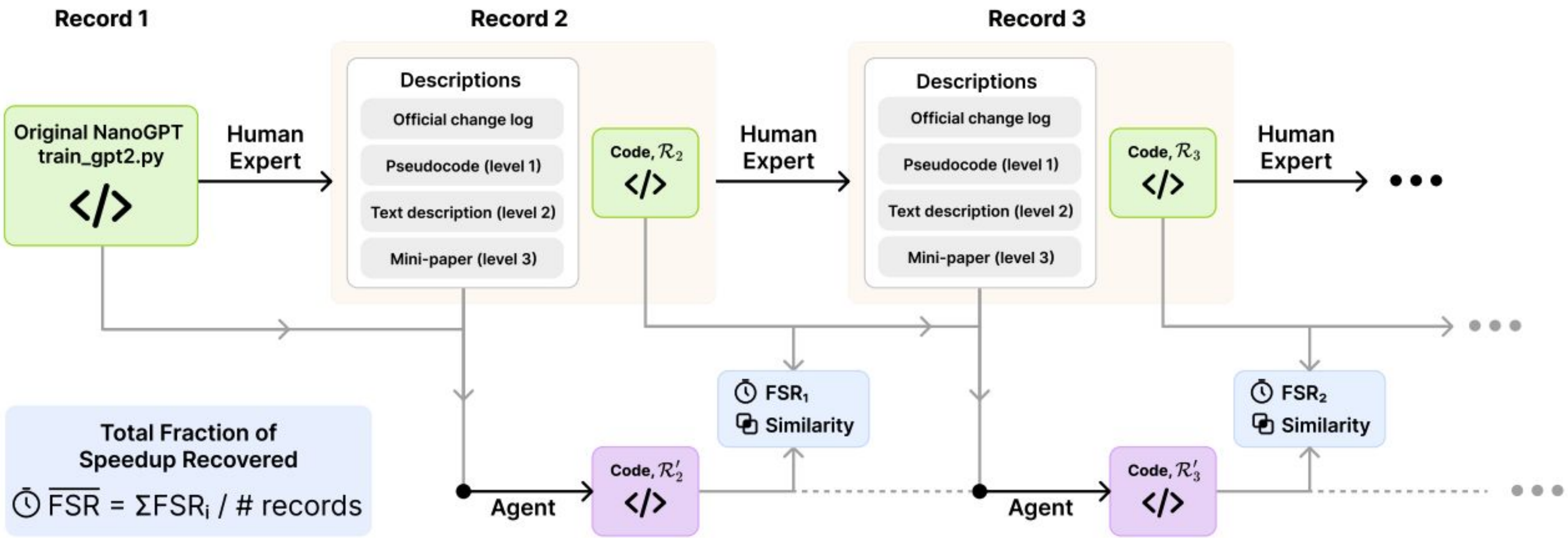


Figure 2 The Automated LLM Speedrunning Benchmark. We create a task for each consecutive pair of records $\mathcal{R}_i$, $\mathcal{R}_{i+1}$. The performance of the agent is evaluated by comparing the relative speedup of the agent solution $\mathcal{R}_i'$ to $\mathcal{R}_i$.

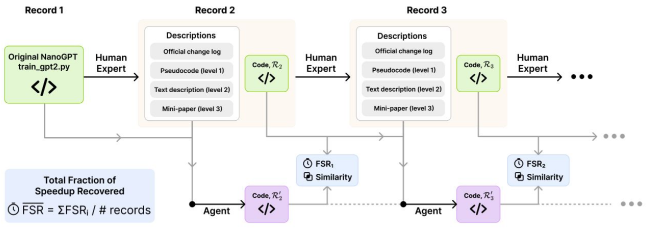


Figure 2 The Automated LLM Speedrunning Benchmark. We create a task for each consecutive pair of records $\mathcal{R}_i$, $\mathcal{R}_{i+1}$. The performance of the agent is evaluated by comparing the relative speedup of the agent solution $\mathcal{R}'_i$ to $\mathcal{R}_i$.

Benchmark

$\mathcal{R}_i$ 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.

- **Record reproduction**: given a set of hints *m* which is any subset of the *hint levels* {1, 2, 3}, can an AI research agent reproduce the speedup from the record?

- **Record optimization**: given no hints *m* = {0}, how does an AI research agent's record improvement trajectory compare to humans?

- **Metric**: *fraction of speedup recovered* (FSR).

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

Level 1 - Pseudo code

```
# Pseudo Code Changes

1. Enhanced Attention Mechanism:
```python
# Replace standard attention with flexible block attention
def flex_attention(q, k, v, block_mask):
    """
    Utilizes blocked sparse attention pattern with:
    - Causal masking (only attend to previous tokens)
    - Document boundary masking (only attend within same document)
    - Sliding window (1024 token context window)
    """
    return optimized_attention(q, k, v, block_mask)

# Generate attention mask with multiple constraints
def create_block_mask(seq_len):
    mask = causal_mask & document_mask & window_mask
    return blocked_sparse_pattern(mask)
```

2. UNet-style Architecture Modifications:
```python
class GPT:
```

Level 2 - Description

```
1. **Specific Improvements Made:**

  - **FlexAttention Implementation:** Replaced standard scaled
  - **Dynamic Block Masking:** Added document-aware causal mask
    - Standard causal attention
    - Document boundary preservation
    - 1024-token sliding window
  - **Sequence Length Expansion:** Increased context length fro
  - **Data Loading Optimization:** Modified DistributedDataLoad
    - Better handle long sequences
    - Reduce document splitting
    - Improve shard management
  - **Memory Efficiency:** Implemented block-wise attention com
  - **Training Optimization:** Adjusted hyperparameters for lar
    - Reduced global batch size from 512 to 8
    - Increased per-device sequence length 64x
    - Adjusted iteration counts
```

Level 3 - Paper

```
# Efficient Training of Large Context Language Models vi

## Abstract
We present architectural and optimization improvements e

## 1. Introduction

### 1.1 Context Length Challenges
Traditional transformer architectures face quadratic mem

- Sparse attention patterns preserving document boundari
- Sliding window attention with 1K local context
- Hardware-aware mask compilation

### 1.2 Key Innovations
```
```

Agent Scaffolds

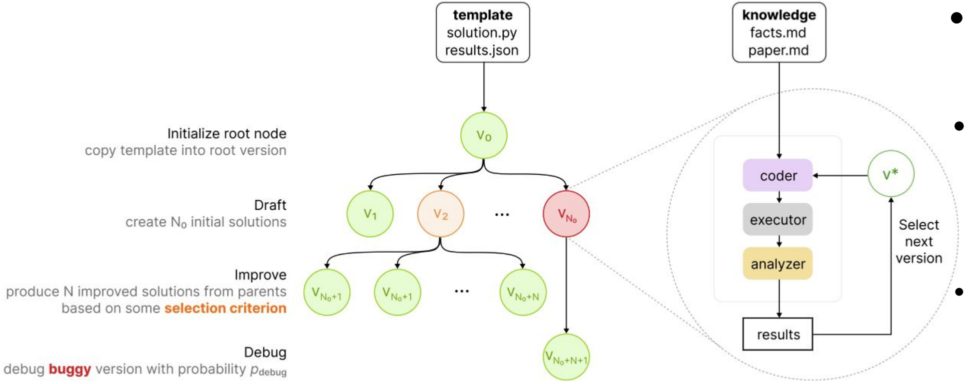


Figure 3 Overview of our flexible search scaffold. Search starts from a root node containing code for the starting record $\mathcal{R}_i$ from which N_0 initial solutions are generated. Subsequently, each search iteration debugs a buggy leaf node with probability p_{debug} and otherwise greedily selects 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).

- **AI research agent**: specific LLM + search scaffold.
- **Search scaffold**: programs that iteratively make use of an LLM for finding a solution to a given task.
- Each search step follows three stages: implementation, execution, analysis. A new node is branched from either a randomly chosen buggy node or the highest-performing node.
- For each scaffold, we use the same budget of 20 steps.

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	—	—	—

How Do Current Agents Fare?

Reproducing individual records

- Without hints, agents fail to recover more than 20% of human speedups.
- Pseudocode (level 1) and pseudocode combinations are the most effective hints.
- Multi-AIDE outperforms other search scaffolds.

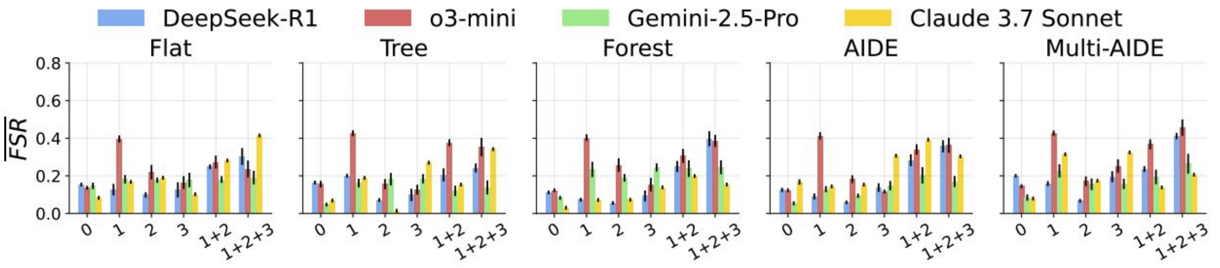


Figure 4 Mean FSR across five search variants and four frontier models for six hint regimes: no hint (0), pseudocode (1), text (2), mini-paper (3) and combinations thereof ($1 + 2$, $1 + 2 + 3$).

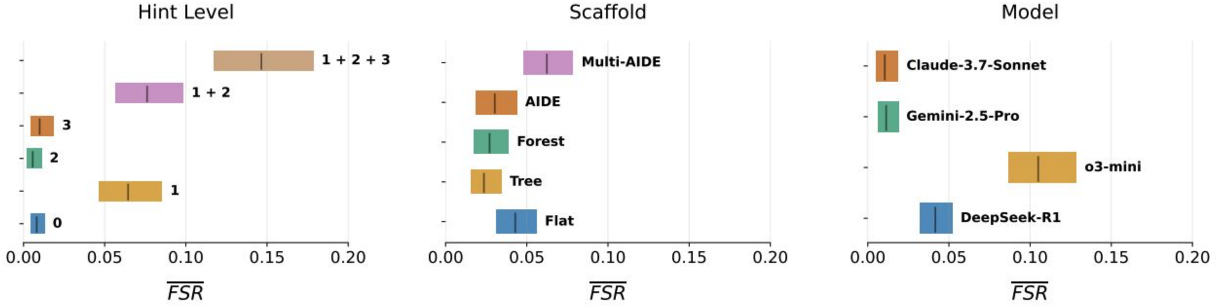


Figure 5 Interquartile Mean (IQM) evaluation results. Scores are aggregated across multiple runs with the same hint level, scaffold, and model.

Cumulative speedrun

- Used best model (o3-mini) with best search scaffold (multi-AIDE).
- Performance significantly drops.

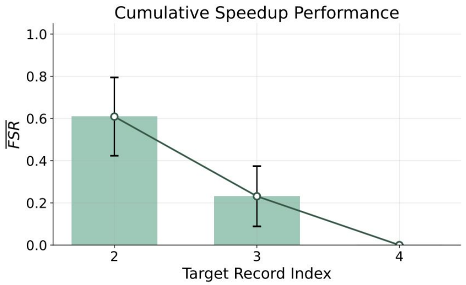


Figure 9 Cumulative Speedup from initial codebase.

LLMs exhibit subtle differences in more granular behaviors.

Search tree composition

- R1 generates more buggy nodes under AIDE/multi-AIDE.
- Gemini-2.5-Pro tends to produce fewer buggy nodes, but it lags behind on FSR metric.
- Claude-3.7-Sonnet generates the most buggy nodes with the fraction increasing over time.

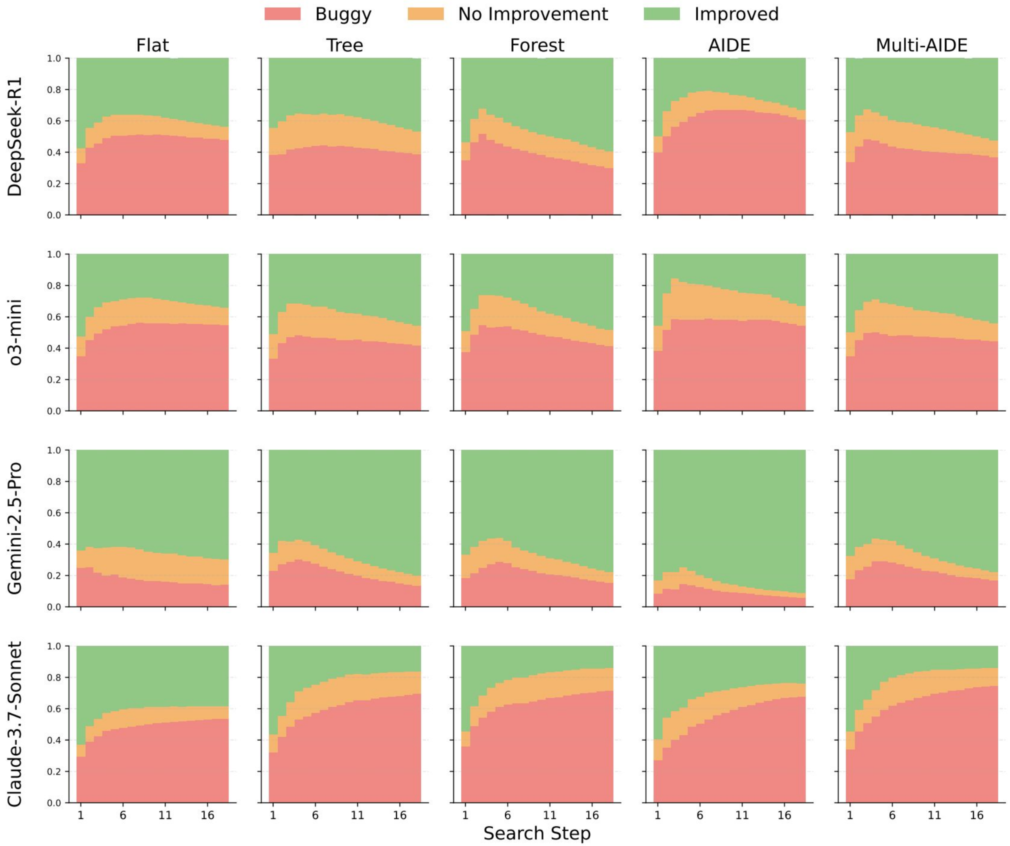


Figure 8 Fraction of node types across search trees for each model and search method. Notably, branching (i.e. non-flat) search is beneficial for reducing the proportion of buggy nodes. Further, a majority of non-buggy steps produce improved nodes for all branching search methods, with the notable exception of Claude-3.7-Sonnet.

How we know agents are faithfully reproducing the target code changes (and not unrelated solutions)?

Embedding based similarity

- Compare code embedding distances between agent and human solution.
- Positive correlation between higher similarity score and FSR for richer hint formats.

LLM-as-a-judge based similarity

- Use a LLM as a judge (R1), prompting it to assess what fraction of the ground-truth changes were successfully reproduced.
- Positive correlation between higher similarity scores and FSR.

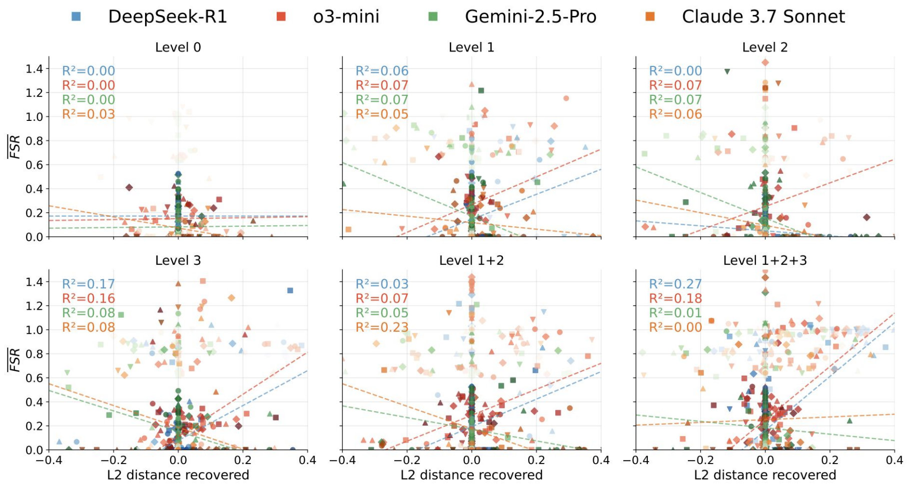


Figure 7 Correlation of FSR with L2 distance recovered for each hint level, showing a modest correlation between similarity to the human solution and FSR for most hint types and models.

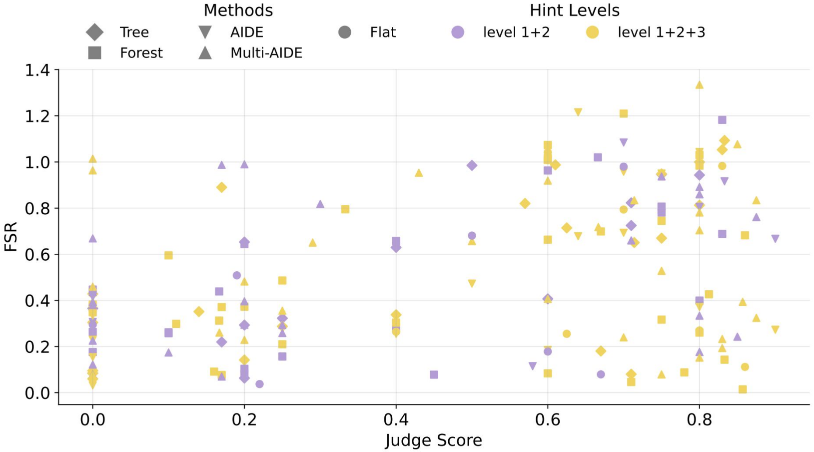
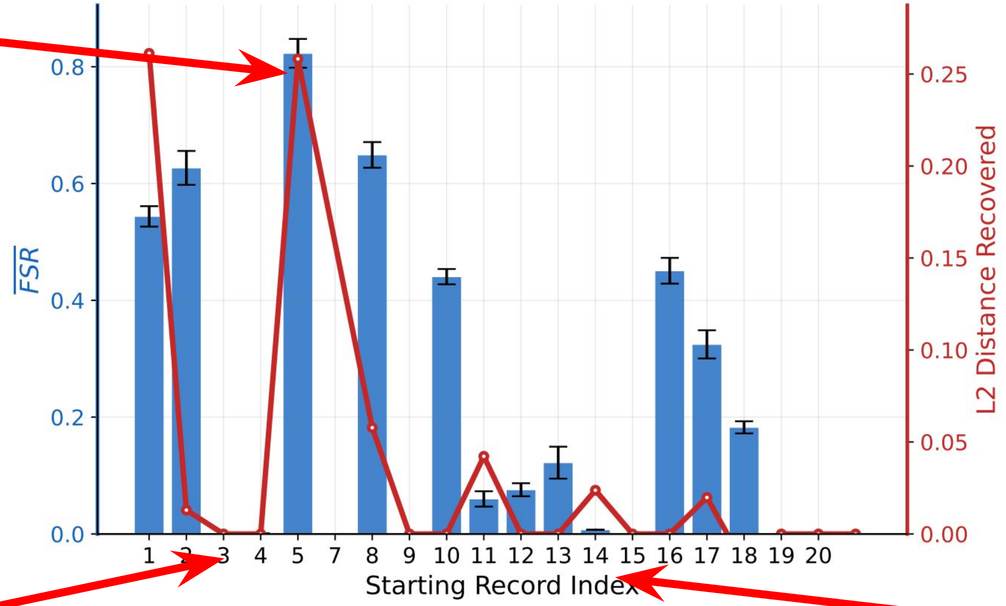


Figure C.2 How FSR (per record) correlates with LLM judge scores for o3-mini-based agents, where a higher judge score means the agent solution is closer to the corresponding human speedrun record.

Record 5 is smallest code changes. ReLU - ReLU^2



Record 3 is Muon Optimizer

Record 12-14 uses FlexAttention API which is outside the knowledge of LLMs.

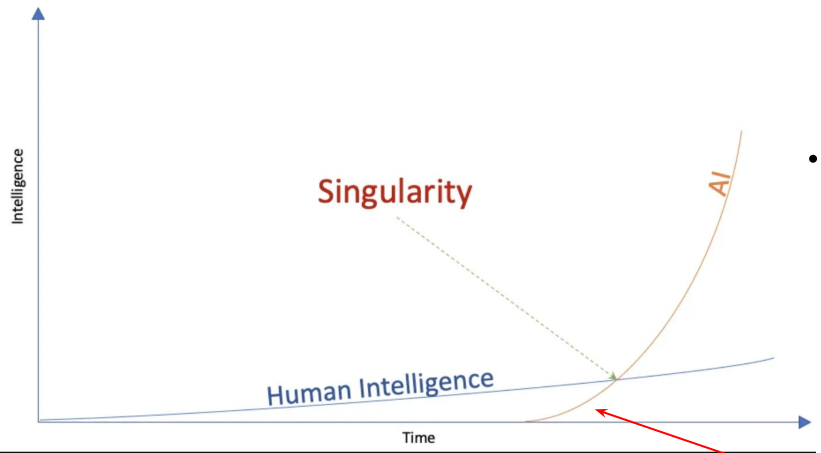
Figure 6 FSR and embedding distance per record for o3-mini with text description hints (mean and std over 3 seeds). Later records tend to be harder for agents, leading to lower recovered embedding distance and speedups.

Summary

- Overall there remain **large gaps** in the ability of AI research agents to reproduce human research innovations even when given detailed hints (e.g. pseudocode), a crucial capability towards the path of automated science.

- The Automated LLM Speedrunning Benchmark is a **challenging and flexible** evaluation that can measure progress towards automated reproducibility by reproducing incremental advances across a chain of research innovations.

- Models exhibit different behaviors/failure modes and with the analysis tools associated with the benchmark, we can better understand more granular model behavior and avenues for improvement.



So maybe we are here