AIRA$_2$: Overcoming Bottlenecks in AI Research Agents

Karen Hambardzumyan[1, 2, *], **Nicolas Baldwin**[1, *], **Edan Toledo**[1, 2, *], **Rishi Hazra**[1, *], **Michael Kuchnik**[1, *], **Bassel Al Omari**[1, 3], **Thomas Simon Foster**[1, 3], **Anton Protopopov**[1], **Jean-Christophe Gagnon-Audet**[1], **Ishita Mediratta**[1], **Kelvin Niu**[1], **Michael Shvartsman**[1], **Alisia Lupidi**[1, 3], **Alexis Audran-Reiss**[1], **Parth Pathak**[1], **Tatiana Shavrina**[1], **Despoina Magka**[1], **Hela Momand**[1], **Derek Dunfield**[1], **Nicola Cancedda**[1], **Pontus Stenetorp**[2], **Carole-Jean Wu**[1], **Jakob Nicolaus Foerster**[1, 3], **Yoram Bachrach**[1], **Martin Josifoski**[1, *]

[1]FAIR at Meta [2]University College London [3]University of Oxford
[*]Equal contribution (author order determined by Mario Kart placement)

Abstract

Existing research has identified three structural *performance bottlenecks* in AI research agents: (1) synchronous single-GPU execution constrains *sample throughput*, limiting the benefit of search; (2) a *generalization gap* where validation-based selection causes overfitting and performance to degrade over extended search horizons; and (3) the *limited capability* of fixed, single-turn LLM operators imposes a ceiling on search performance. We introduce AIRA$_2$, which addresses these bottlenecks through three architectural choices: an asynchronous multi-GPU worker pool that increases experiment throughput linearly; a Hidden Consistent Evaluation protocol that delivers a reliable evaluation signal; and ReAct agents that dynamically scope their actions and debug interactively. On MLE-bench-30, AIRA$_2^\dagger$ achieves a mean Percentile Rank of **81.5%** at 24 hours and **83.1%** at 72 hours, outperforming the strongest baseline, which achieves 72.7%. On AIRS-Bench, AIRA$_2$ exceeds human state-of-the-art on 6 out of 20 diverse research tasks. Ablations confirm that each architectural component is necessary, that performance follows a predictable scaling law that transfers across LLM backbones, and that the "overfitting" reported in prior work was driven by evaluation noise rather than true data memorization.

1 Introduction

The rapid advancement of Large Language Model (LLM) capabilities has enabled autonomous agents capable of executing complex, multi-step workflows [Josifoski et al., 2023, Wu et al., 2024].

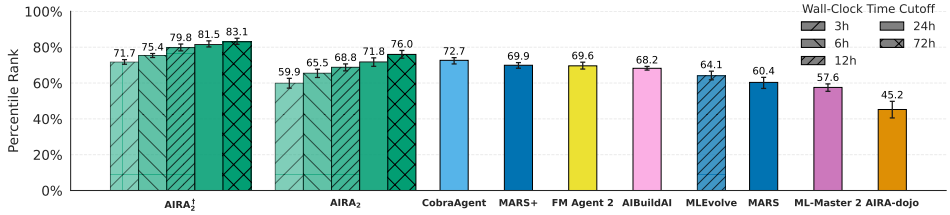


Figure 1: **AIRA$_2$ performance on MLE-bench-30.** AIRA$_2$ achieves the highest Percentile Rank among the systems shown and improves consistently with additional compute, demonstrating the effectiveness of our architectural design. The other agents are drawn from the public leaderboard and may differ from AIRA$_2$ in LLM backbone and search-time compute (GPU-hours), and may employ additional augmentations such as runtime web retrieval or task-specific knowledge bases that AIRA$_2$ does not use; they are shown for reference only, and we isolate the effect of each factor in our controlled ablations (Section 4.3). AIRA$_2^\dagger$ uses Gemini 3.1, while AIRA$_2$ uses Gemini 3.0.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

While these agents have achieved remarkable success in software engineering [Wang et al., 2025b, Anthropic, 2025, OpenAI, 2025] and mathematics [Novikov et al., 2025, Hubert et al., 2025]—benefiting from execution environments with rapid feedback loops of verifiable signal—automating the scientific process presents a distinct class of challenges. Scientific research requires agents to learn through controlled experimentation within a vast, partially observable, open-ended design space [Langley et al., 1987], navigating noisy evaluation signals and high-latency compute-intensive experiments [Chan et al., 2025].

Prior work, AIRA-dojo [Toledo et al., 2025], frames research agents as a search over candidate solutions and formalizes three structural bottlenecks preventing further scaling: **(1) Compute Throughput**, **(2) Generalization Gap**, and **(3) Operator Capability**. As noted by Chan et al. [2025], Jiang et al. [2025], Zhu et al. [2026], these performance plateaus emerge even within a relatively short 24-hour regime, suggesting that addressing them is a prerequisite for effectively utilizing additional compute. We introduce AIRA_2, a research agent *designed to overcome these bottlenecks* via three architectural choices:

First, synchronous single-GPU execution severely bottlenecks throughput: the search process stalls whenever an expensive experiment runs, starving exploration within the available wall-clock budget. AIRA_2 *introduces an asynchronous multi-GPU worker pool* that decouples decision-making from execution, increasing experiment throughput linearly with available GPU resources. Concretely, 8 GPUs yield approximately $8\times$ the experimental throughput, compressing what would otherwise require days of sequential exploration into hours.

Second, the generalization gap undermines long-horizon search: agents optimize validation metrics at the expense of held-out test performance, causing trajectories to overfit. AIRA_2 *addresses this with a Hidden Consistent Evaluation (HCE) protocol*: data splits are standardized once and reused across all candidates, evaluation labels are hidden from the agent, and the search signal is decoupled from the final selection signal. HCE enables continued performance gains with additional compute.

Third, operators that depend on fixed prompts and single-turn actions impose a performance ceiling: for example, a static "debug" prompt cannot iteratively diagnose complex errors, and no amount of search sophistication can compensate. AIRA_2 *replaces all operators with ReAct agents* [Yao et al., 2023] that autonomously scope their actions—performing exploratory data analysis, running development experiments, inspecting logs, and more—expanding the range of tasks the system can tackle and speeding up the discovery of strong solutions.

Together, these design decisions enable AIRA_2 to achieve a mean Percentile Rank of **81.5%** at 24 hours and **83.1%** at 72 hours on MLE-bench-30 [Singh et al., 2025], outperforming the strongest baseline, which achieves 72.7%. Beyond the Kaggle-style competitions of MLE-bench, we evaluate AIRA_2 in a case study on AIRS-Bench [Lupidi et al., 2026], where it exceeds the published state-of-the-art on 6 out of 20 diverse research tasks. Ablations confirm that each architectural component contributes to performance, and that performance follows a predictable scaling law that transfers across LLM backbones. $\text{AIRA}_2^\dagger$ denotes the use of Gemini 3.1, while all other AIRA_2 variants in the paper use Gemini 3.0. The consistent gains when using a stronger model backbone confirm that the architecture scales with model capability.

Our contribution is primarily empirical: we provide the first controlled, module-by-module decomposition of large-scale agent search performance. Each configuration is evaluated with 8 GPUs for 72 hours on 30 tasks with 3 seeds ($8 \times 72 \times 30 \times 3 = 51{,}840$ GPU-hours), and the full study totals over **324,000 GPU-hours** of experimentation. Given the recent trend of large-scale parallelized agents [Novikov et al., 2025, Yamada et al., 2025, Wen et al., 2026] which are rapidly becoming the norm, we believe that a principled empirical understanding of what drives performance is exactly what the field needs.

2 Background and Related Work

The domain of automated machine learning has shifted from simple heuristics [Bergstra and Bengio, 2012, Elsken et al., 2017, Li et al., 2018] to autonomous agents capable of long-horizon research [Yamada et al., 2025]. LLM-powered agents now tackle open-ended tasks in software engineering [Wang et al., 2025b, Anthropic, 2025, OpenAI, 2025], mathematics [Novikov et al., 2025, Hubert et al., 2025], chemistry [Wang et al., 2025a], material science [Abhyankar et al., 2025], and computational

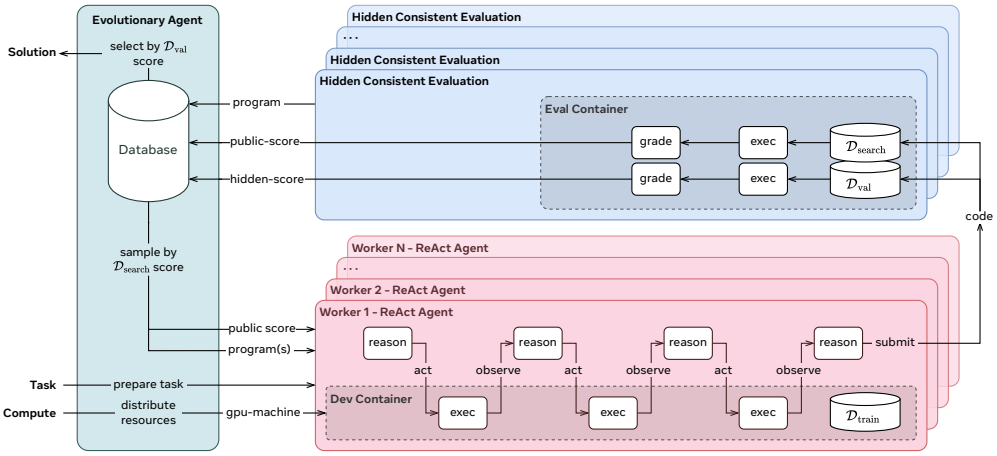


Figure 2: **AIRA$_2$ architecture.** The Evolutionary Agent maintains a population of candidate solutions and dispatches mutation tasks to N workers as they become available. Each worker *asynchronously* executes a ReAct agent which iteratively reasons, executes code, and observes outputs. Candidate solutions are evaluated in a separate container; agents observe only the resulting score. Evaluation is partitioned: $\mathcal{D}_{\text{search}}$ guides optimization while $\mathcal{D}_{\text{val}}$ determines final selection.

complexity theory [Yu et al., 2025]. For AI research (**AI4AI**), recent agents achieve strong benchmark performance by scaling inference-time compute via tree [Du et al., 2025, Chen et al., 2026] or graph search [Botla et al., 2025, Li et al., 2025], multi-agent collaboration [Gottweis et al., 2025], advanced context management [Liu et al., 2025b], and external knowledge [Nam et al., 2025].

However, these systems often conflate multiple design choices, making it difficult to isolate which components drive performance. AIRA-dojo [Toledo et al., 2025] formalises a systematic approach by disentangling gains from infrastructure, operators, and search strategy, exposing three structural bottlenecks: **(1) Compute throughput**—synchronous single-GPU execution constrains sample generation, limiting exploration to ≈1–20 candidates per day on compute-heavy tasks; **(2) Generalization gap**—agents "game" self-reported metrics and reuse validation sets for both hill-climbing and selection, with oracle experiments showing that test-based selection improves medal rates by 9–13% (absolute); and **(3) Static operators**—fixed, single-turn prompts cannot adapt to unpredictable dependencies, and more advanced search yields no significant improvement over greedy baselines when operators are static. We build on this decomposition by targeting each bottleneck with a dedicated architectural module, providing controlled ablations that attribute performance to individual components. Relative to similar evolutionary approaches [Novikov et al., 2025, Lange et al., 2026], we provide a module-by-module ablation of the end-to-end research agent.

3 AIRA$_2$ Research Agent Design

In this section, we describe the design of AIRA$_2$, a system that addresses the three bottlenecks identified in Section 2. Its architecture comprises two tiers: a **Global Orchestrator** that algorithmically coordinates search over a population of candidates, and an **Asynchronous Worker Pool of ReAct agents** mutating solutions in isolated containers (Figure 2).

3.1 Evolutionary Search

The orchestrator maintains a population $\mathcal{P}$ of candidate solutions, their fitness scores, and all other associated metadata in a graph database. Search proceeds via asynchronous, steady-state evolution [Syswerda, 1991]: whenever a worker becomes available, the orchestrator samples a parent (or two) and dispatches a mutation/crossover task.

Selection. Parents are sampled via temperature-scaled rank-based selection. Given a population $\mathcal{P}$ of size N sorted by fitness, we assign rank $r_i \in \{1, \ldots, N\}$ to each individual (where 1 is the best).

The probability of selecting individual i is:

$$p(i) = \frac{(N - r_i + 1)^{1/T}}{\sum_{j=1}^{N}(N - r_j + 1)^{1/T}}, \tag{1}$$

where T controls the exploration-exploitation tradeoff. We opted for rank-based rather than fitness-proportionate selection because ranks are invariant to the magnitude and scale of fitness scores, which vary widely across tasks.

Mutation/Crossover. The orchestrator randomly selects between *mutation* (refining a single parent) and *crossover* (combining two parents) with probability c. Both operations are executed by ReAct agents (Section 3.4), which receive the parent solution(s) (and other metadata) as context. The resulting program is evaluated by the orchestrator and added to the population $\mathcal{P}$ with its fitness score.

3.2 Scaling Compute: Asynchronous Multi-GPU Execution

The asynchronous nature of steady-state evolution inherently facilitates parallel orchestration, allowing us to distribute the search workload across concurrent workers without synchronization barriers. Unlike generational evolution, which must wait for all workers to complete before proceeding, steady-state evolution dispatches mutations as soon as any worker becomes available, ensuring fast-completing workers are never left idle. Each worker is assigned a dedicated GPU with an identical, ephemeral `Apptainer` [Kurtzer et al., 2021] container using the `Superimage` environment [Toledo et al., 2025], which provides a comprehensive suite of ML packages and `fakeroot` privileges for dependency installation. This ensures reproducible, isolated execution where crashed containers do not affect the orchestrator or other workers. Additional implementation details (resource allocation, data management, stateful tools) are provided in Appendix A.

3.3 Closing the Generalization Gap: Hidden Consistent Evaluation

To close the generalization gap, AIRA_2 decouples the signal used for search from the signal used for final selection, and externalizes all evaluation to prevent metric gaming. Beyond serving as a practical safeguard, this protocol is designed as an experimental tool: by controlling the evaluation procedure, we can systematically test whether agents truly overfit to their data or whether previously reported degradation stems from evaluation noise (Section 4.3.2).

Data partitioning. Before search begins, available labels are split into three disjoint sets:

- $\mathcal{D}_{\text{train}}$: visible to the agent for model training.
- $\mathcal{D}_{\text{search}}$: used by the orchestrator for fitness computation; *labels hidden from agents*.
- $\mathcal{D}_{\text{val}}$: used *only* for final selection after search terminates; *hidden from both agents and the search process.*

These splits are created once via random sampling of the available labeled training data (80%/10%/10%) and reused identically across all seeds. Notably, baselines are evaluated using their published results on the $\mathcal{D}_{\text{test}}$ (official test set) with access to all training data, while AIRA_2 reserves 20% of training data for search/selection, making the comparison conservative. AIRA_2 does not retrain on the full available data for the results reported in this work. Upon submission, the solution submits predictions for $\mathcal{D}_{\text{test}}$. Thus, all splits are consistent across programs and seeds.

Externalized evaluation. Agents never self-report metrics. When a worker returns a solution, the orchestrator evaluates it on $\mathcal{D}_{\text{search}}$ in a separate container, mirroring the dev environment. Agents observe only the resulting score, not the labels, preventing feedback loops that enable metric hacking.

Decoupled selection. Because $\mathcal{D}_{\text{val}}$ is never used during search, the final selection is insulated from hill-climbing dynamics. This separation—combined with hidden consistent evaluation across all candidates—is designed to reduce the validation–test gap observed in prior systems.

Table 1: **Main performance evaluation across varying time budgets (3h, 24h, 72h).** We report Percentile Rank and medal rates for AIRA_2 (8 GPUs, Subagents, HCE) against ablations and baselines. ±SE; $\text{AIRA}_2^\dagger$ and Mini-SWE-Agent† use Gemini 3.1 Pro, all other AIRA_2 variants Gemini 3.0.

	% Percentile Rank			% Bronze+			% Silver+			% Gold		
Method	3h	24h	72h	3h	24h	72h	3h	24h	72h	3h	24h	72h
$\text{AIRA}_2^\dagger$	**71.7 ±0.8**	**81.5 ±1.1**	**83.1 ±1.1**	**54.4 ±2.9**	**72.2 ±1.9**	**76.7 ±1.6**	**47.8 ±2.5**	**65.6 ±2.7**	**73.3 ±2.2**	**28.9 ±1.9**	**41.1 ±2.5**	**52.2 ±2.7**
$\text{AIRA}_2^\dagger$ (1 GPU)	39.0 ±3.1	69.4 ±1.1	73.7 ±1.8	27.8 ±3.1	55.6 ±2.5	62.2 ±2.5	23.3 ±3.1	53.3 ±2.9	62.2 ±2.5	14.4 ±2.7	28.9 ±2.5	32.2 ±1.9
AIRA_2	59.9 ±1.6	71.8 ±1.5	76.0 ±1.3	38.9 ±2.2	57.8 ±2.7	61.1 ±2.7	33.3 ±2.5	50.0 ±2.7	58.9 ±2.5	20.0 ±2.2	32.2 ±2.7	36.7 ±2.2
AIRA_2 (4 GPU)	56.9 ±1.8	71.2 ±1.5	76.5 ±1.2	37.8 ±3.1	55.6 ±3.1	62.2 ±2.5	30.0 ±2.5	47.8 ±3.1	55.6 ±2.5	11.1 ±1.9	30.0 ±2.2	40.0 ±2.5
AIRA_2 (2 GPU)	51.8 ±1.8	64.4 ±1.3	71.0 ±1.2	35.6 ±2.7	46.7 ±3.1	56.7 ±2.5	30.0 ±2.9	43.3 ±3.1	54.4 ±2.9	14.4 ±1.9	20.0 ±2.5	31.1 ±2.7
AIRA_2 (1 GPU)	41.3 ±2.1	56.8 ±1.1	63.5 ±1.2	22.2 ±2.2	41.1 ±2.5	51.1 ±2.7	17.8 ±1.9	32.2 ±2.2	41.1 ±2.5	10.0 ±1.6	20.0 ±1.9	24.4 ±2.7
AIRA_2 (No Subagents)	54.4 ±1.2	68.6 ±1.5	73.7 ±1.3	33.3 ±2.5	52.2 ±2.9	60.0 ±2.9	30.0 ±1.6	47.8 ±2.2	53.3 ±2.9	13.3 ±2.5	32.2 ±2.7	37.8 ±1.1
AIRA_2 (No HCE)	44.9 ±2.7	58.8 ±2.5	57.6 ±2.4	32.8 ±3.5	50.3 ±2.8	51.1 ±2.9	28.6 ±3.4	49.2 ±2.8	50.0 ±2.9	14.4 ±2.5	33.3 ±3.2	35.6 ±2.8
AIRA_2 (No Evo.)	55.4 ±1.0	64.3 ±1.3	65.9 ±1.3	36.7 ±1.6	46.7 ±1.9	47.8 ±1.6	32.2 ±1.9	40.0 ±2.5	44.4 ±2.2	16.7 ±1.6	24.4 ±1.9	24.4 ±1.9
$\text{AIRA}_2^\dagger$ (1 GPU) with Mini-SWE-Agent	49.6 ±1.6	64.4 ±1.7	72.6 ±1.7	31.1 ±2.5	47.8 ±3.1	57.8 ±3.1	24.4 ±2.7	41.1 ±3.3	51.1 ±2.9	13.3 ±1.9	24.4 ±2.5	33.3 ±2.5
CobraAgent [Dalpha Team, 2026] Gemini 3.1 Pro + Flash	72.7 ±1.1			**78.9 ±1.1**			53.3 ±4.0			16.7 ±2.2		
MARS+ [Chen et al., 2026] Gemini 3.0 Pro	69.9 ±1.0			64.4 ±1.6			51.1 ±2.7			24.4 ±2.5		
FM-Agent 2.0 [Li et al., 2025] Gemini 3.0 Pro	69.6 ±1.2			61.1 ±1.9			57.8 ±2.2			36.7 ±3.1		
AIBuildAI [Zhang et al., 2026] Claude Opus 4.6	68.2 ±0.7			64.4 ±1.1			42.2 ±3.3			12.2 ±1.9		
MLEvolve [Du et al., 2025] Gemini 3.0 Pro	64.1 ±1.6			57.8 ±2.5			52.2 ±2.2			22.2 ±2.5		
MARS [Chen et al., 2026] Gemini 3.0 Pro	60.4 ±1.9			54.4 ±2.9			44.4 ±3.5			18.9 ±1.9		
ML-Master 2.0 [Liu et al., 2025b] DeepSeek V3.2	57.6 ±1.3			52.2 ±2.2			40.0 ±3.1			8.9 ±1.1		
PiEvolve [Botla et al., 2025] Gemini 3.0 Pro	54.1 ±1.2			54.4 ±1.1			50.0 ±1.6			27.8 ±3.3		
AIRA-dojo [Toledo et al., 2025] Gemini 3.0 Pro	45.2 ±2.9			27.8 ±3.8			24.4 ±3.7			14.4 ±2.7		

3.4 Beyond Static Operators: ReAct Agents

To overcome the operator limitations, AIRA_2 replaces static operators with ReAct agents [Yao et al., 2023] that execute multi-step reasoning trajectories. Each mutation produces a trajectory:

$$\tau = \big(\text{Reason}_1, \text{Act}_1, \text{Obs}_1, \ldots, \text{Reason}_{K-1}, \text{Act}_{K-1}, \text{Obs}_{K-1}, \text{Reason}_K, \text{Act}_K\big), \tag{2}$$

where *Actions* (Act_t) are Python or Bash commands executed in a sandboxed environment, and *Observations* (Obs_t) consist of execution outputs. The final Act_K is the "submit" tool, which sends the solution back to the orchestrator for evaluation.

This formulation provides two capabilities that fixed operators lack:

Dynamic scoping. The agent decides at runtime what actions are necessary—inspecting data distributions before modelling, or prioritizing hyperparameter tuning for refinement tasks—eliminating brittle "scope engineering" in fixed operator prompts and significantly broadening the action space.

Interactive debugging. When code raises an exception, the agent observes the traceback within the same trajectory, hypothesizes a fix, and re-executes—resolving errors without forfeiting the mutation attempt.

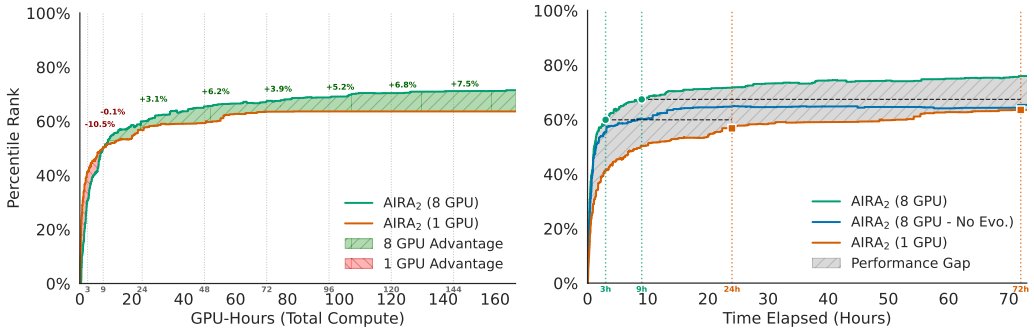


Figure 3: **(a) Compute Efficiency.** 8 GPUs incur an initial exploration cost but yield superior long-term performance, with the gap widening to 7.5 Percentile Rank points at 144 GPU-hours. **(b) Search Strategy at Scale.** Best-of-K (8 GPUs, no evolution) saturates at the same level as the 1-GPU evolutionary agent. Effective use of parallel compute requires both additional resources and evolutionary selection.

4 Experiments and Results

4.1 Experimental Setup

Tasks. We evaluate AIRA$_2$ on MLE-bench-30, a curated subset of 30 Kaggle competitions from MLE-bench [Chan et al., 2025] used in the GPT-5 system card [Singh et al., 2025]. We opted for this subset to facilitate a lightweight yet representative evaluation; unlike MLE-bench Lite, which focuses primarily on low-complexity tasks, MLE-bench-30 spans a broader difficulty spectrum stratified into 5 low, 20 medium, and 5 high complexity tasks. Following standard protocol, we report the **medal rate**—the fraction of runs achieving at least a bronze medal on the Kaggle leaderboard. However, our primary analytical metric is **Percentile Rank**, representing the agent's simulated leaderboard position. Calculated as $P = \frac{N-R}{N-1} \times 100$ (where N is total entries in the snapshot of leaderboard and R is the ordinal rank, with 1 being best), Percentile Rank offers three advantages over medal rate: (1) it is continuous rather than discrete, enabling finer-grained comparisons; (2) it captures the full distribution of performance rather than a binary medal outcome; and (3) it avoids threshold effects near medal boundaries that amplify noise in aggregate statistics [Audran-Reiss et al., 2025]. Importantly, gains at higher percentiles are progressively harder to achieve, as each increment requires outperforming increasingly skilled human competitors.

System configuration. While AIRA$_2$ can scale to large GPU counts, for these experiments, we use 8× `NVIDIA H200 GPUs` (`141GB VRAM` each) with 1:1 worker-to-GPU mapping. Workers execute in `Apptainer` containers with `CUDA`, `PyTorch`, and standard data science libraries. ReAct agents are powered by `Gemini 3.0 Pro Preview` [Google DeepMind, 2025] in all AIRA$_2$ variants, with the exception of AIRA$_2^\dagger$, which shares the same architecture but uses `Gemini 3.1 Pro Preview` as the LLM backbone. The orchestrator runs steady-state evolution with temperature-scaled rank selection ($T = 0.2$) and crossover probability $p_c = 15\%$.

Protocol. Each task runs for 72 hours wall-clock time. We run 3 independent seeds per task and report the mean over tasks of per-task seed means ± the standard error across seeds with the tasks held fixed, SE $= \frac{1}{30}\sqrt{\sum_t s_t^2/n_t}$, where s_t^2 is the sample variance over the n_t seeds of task t; the same procedure is applied to all rows, including baselines. Task data is partitioned as 80% $\mathcal{D}_{\text{train}}$ / 10% $\mathcal{D}_{\text{search}}$ / 10% $\mathcal{D}_{\text{val}}$. Final submissions are evaluated directly on the held-out test set $\mathcal{D}_{\text{test}}$ **without** retraining on the full available training data $\cup\{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{search}}, \mathcal{D}_{\text{val}}\}$.

4.2 Main Results

Table 1 presents the performance of AIRA$_2$ and AIRA$_2^\dagger$ under varying time budgets and component ablations. Baselines span a range of LLM backbones: MARS+, MARS, MLEvolve, FM-Agent 2.0, PiEvolve, and AIRA-dojo use Gemini 3.0 Pro Preview [Google DeepMind, 2025]; CobraAgent [Dalpha Team, 2026] uses a Gemini 3.1 Pro Preview and Gemini Flash in an ensemble; AIBuildAI [Zhang et al., 2026] uses Claude Opus 4.6; and ML-Master 2.0 uses DeepSeek V3.2-Speciale [Liu et al., 2025a]. We note that CobraAgent, AIBuildAI, MARS+, MARS, FM-Agent 2.0, and MLEvolve are

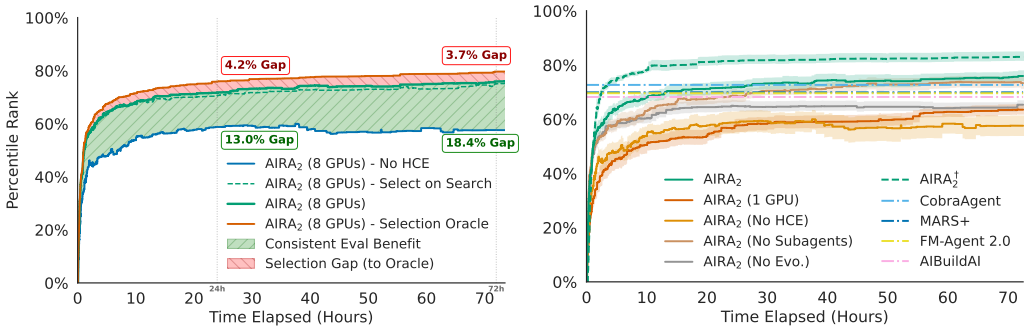


Figure 4: **(a) Stabilizing Long-Horizon Search.** Self-reported evaluation (blue) leads to performance degradation, while Hidden Consistent Evaluation (green) ensures long-term improvement. The marginal difference between selecting via $\mathcal{D}_{\text{search}}$ (seen) and $\mathcal{D}_{\text{val}}$ (unseen) suggests the degradation in prior work was due to evaluation noise, not overfitting. **(b) Performance profile:** Steady increase across all configurations except No HCE, which stagnates, with AIRA$_2^\dagger$ achieving the highest Percentile Rank among all agents.

concurrent work released after the development and evaluation of AIRA$_2$. CobraAgent, MARS+ and ML-Master 2.0 utilize 2 GPUs; all other baselines use a single GPU. We additionally include single GPU runs of Mini-SWE-Agent† [Yang et al., 2024b] (Gemini 3.1) as a reference point with an established agent; to make use of HCE and the evaluation feedback, it is run within our pipeline, in the same execution environment as all AIRA$_2$ variants, for a controlled comparison (see also Appendix B). We report Percentile Rank as our primary metric, as discrete medal thresholds are sensitive to noise near boundaries and fail to capture progress on difficult tasks where no agent achieves medals—for instance, improving from the 5th to the 55th percentile on a hard task is invisible to medal rates but reflected in Percentile Rank.

Early Search (3h) At the 3-hour mark, AIRA$_2^\dagger$ achieves a Percentile Rank of 71.7%, already matching the strongest 24-hour baselines such as CobraAgent (72.7%) and MARS+ (69.9%) within just 24 GPU-hours. The base AIRA$_2$ configuration reaches 59.9%, comparable to top single-GPU agents. While the system is not optimized for compute efficiency—it uses 8 GPUs to prioritize search breadth over resource efficiency—these early results demonstrate that strong initial solutions emerge quickly and provide a foundation for continued refinement.

Standard Evaluation (24h) At 24 hours, AIRA$_2^\dagger$ achieves a mean Percentile Rank of 81.5%, surpassing the strongest baseline, CobraAgent [Dalpha Team, 2026], by 8.8 percentage points. The base AIRA$_2$ configuration reaches 71.8%, competitive with the top baselines; notably, a 4-subagent configuration (each with a dedicated GPU) achieves 71.2%—within 0.6 percentage points—indicating that competitive performance does not require the full 8-subagent setup (see Section 4.4). Both results are achieved despite reserving 20% of the training data for the Hidden Consistent Evaluation protocol (Section 3.3), which trades short-term performance for a more reliable search signal.

Long-Horizon Search (72h) AIRA$_2$ is designed for sustained improvement over extended time horizons. At 72 hours, AIRA$_2^\dagger$ reaches 83.1% Percentile Rank and AIRA$_2$ reaches 76.0%—both surpassing all reported 24-hour baselines. Notably, unlike prior systems where extended runtime leads to performance degradation due to overfitting [Toledo et al., 2025], both configurations continue to improve with additional compute. The consistent gains when using a stronger model backbone confirm that the architecture scales with model capability. We analyse the specific contributions of each ablated component (Subagents, Consistent Eval) in Section 4.3.

In Appendix H, we present detailed case studies including trajectory-level analysis of "eureka moments" on complex MLE-bench tasks where AIRA$_2$ uniquely surpassed medal thresholds. We also evaluate AIRA$_2$ on AIRS-Bench [Lupidi et al., 2026], a suite of 20 research tasks spanning molecular property prediction, time series forecasting, NLP, and more: AIRA$_2$ exceeds the published human state-of-the-art on 6 tasks using clean, inductive methodologies and comes within 95% of SOTA on several others.

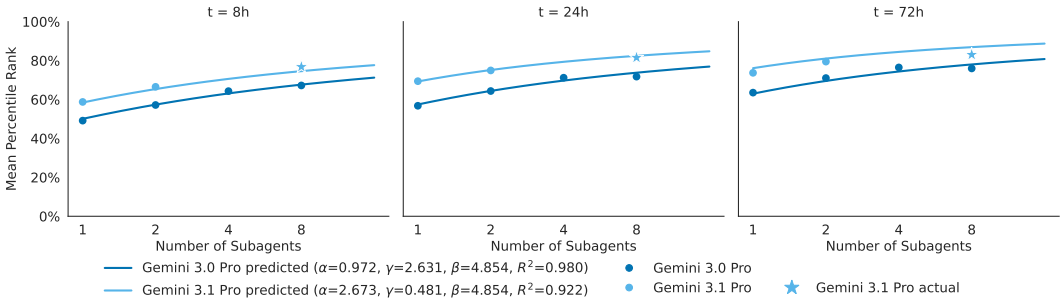


Figure 5: **Predictable performance scaling across models.** The fitted scaling law captures the diminishing returns of subagent interaction over time (Equation 3), tightly fitting the Gemini 3.0 Pro baseline ($R^2 = 0.98$). Notably, the architecture's scaling parameter transfers across backbones: calibrating Gemini 3.1 Pro on only one or two subagents accurately predicts the performance of the unseen $N = 8$ configuration ($R^2 = 0.92$).

4.3 Ablation Studies

We adopt a subtractive ablation design, removing one component at a time from the full system. All ablations use AIRA_2 (Gemini 3.0 Pro Preview) as the base configuration.

4.3.1 Resolving the Compute Bottleneck

Parallel compute improves efficiency, not just speed. In Figure 3a, we normalize performance by cumulative GPU hours. The 8-GPU setup initially underperforms as it builds a diverse population, but then improves significantly: the gap widens from 3.1 Percentile Rank points at 24 GPU-hours to 7.5 at 144 GPU-hours, indicating that broader parallel exploration prevents local optima traps.

Parallelism without evolution is suboptimal. In Figure 3b, we compare AIRA_2 against a Best-of-K baseline: 8 independent workers, each repeatedly generating and submitting solutions from scratch without shared population or lineage. While Best-of-K scales rapidly in the first few hours, it plateaus at the same level as the single-GPU evolutionary agent, effectively "wasting" 7 GPUs. In contrast, AIRA_2's shared evolutionary population ensures additional throughput translates into higher asymptotic performance.

4.3.2 Closing the Generalization Gap

Without HCE: reproducing degradation. We replicate the setup of Toledo et al. [2025] and Jiang et al. [2025], where agents self-report metrics using dynamic validation splits. Our results confirm their findings: performance peaks early and subsequently degrades (Figure 4a). We hypothesize that this degradation is driven by **evaluation noise**—"lucky" splits and spurious successes create false positive signals that destabilize the search trajectory (see Appendix D for a concrete example).

With HCE: eliminating degradation. Under HCE, the search set $\mathcal{D}_{\text{search}}$ is fixed externally and hidden from the agent, and selection uses a held-out $\mathcal{D}_{\text{val}}$. As shown in Figure 4a, this eliminates the degradation entirely. HCE accounts for a 13.0 Percentile Rank point improvement at 24 hours and 18.4 points at 72 hours, and without HCE performance stagnates between 24h and 72h, confirming that HCE is essential for sustained long-horizon improvement.

Diagnosing the remaining gap: noise, not memorization. If classical overfitting to $\mathcal{D}_{\text{search}}$ were occurring, test performance should degrade over time when selecting on $\mathcal{D}_{\text{search}}$. However, as shown in Figure 4a, test performance improves monotonically even under $\mathcal{D}_{\text{search}}$ selection, and the difference between selecting on $\mathcal{D}_{\text{search}}$ versus the unseen $\mathcal{D}_{\text{val}}$ is marginal. This strongly suggests that the degradation observed in prior work was driven by evaluation noise, not data memorization.

4.3.3 Overcoming the Static Operator Limitation

As shown in Table 1 & Figure 4b, ReAct agents outperform single-turn operators by 5.5 Percentile at 3 hours, acting as an efficiency multiplier for early discovery. The gap narrows to 2.3 points at 72 hours, suggesting that single-turn operators are not hitting a hard ceiling but are inefficient: the

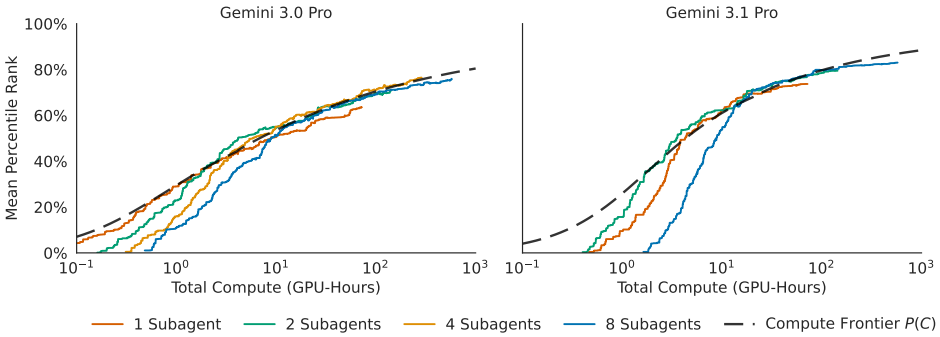


Figure 6: **Predicted compute frontier.** Observed performance across AIRA_2 configurations closely tracks the predicted compute frontier $P(C)$ (dashed curve, Equation 4).

evolutionary loop eventually compensates by externalizing context between independent attempts. We hypothesize that the benefit of ReAct would be more pronounced in scenarios requiring broader tool use (e.g., internet browsing, API interaction), where multi-turn navigation is required.

Replacing our ReAct subagent with an established agent, Mini-SWE-Agent†, yields comparable performance at 72 hours (72.6 vs. 73.7 for $\text{AIRA}_2^\dagger$ (1 GPU); Table 1), although Mini-SWE-Agent† is stronger early on (49.6 vs. 39.0 at 3 hours). At matched GPU-hour budgets, $\text{AIRA}_2^\dagger$ with 8 subagents outperforms both from 24 GPU-hours onward (Appendix B), further confirming the utility of the orchestration layer rather than the specific choice of agent.

4.4 Scaling Laws

A natural question for practitioners is how to allocate compute: given a fixed GPU budget, is it better to run more subagents for less time, or fewer for longer? We find that a simple parametric model captures the relationship between performance P, subagent count N, and wall-clock time t:

$$P(N,t) = 100 \cdot \frac{g(N,t)}{g(N,t)+1}, \qquad g(N,t) = \alpha \cdot \log(\beta\, N + 1) \cdot \log(\gamma\, t + 1), \tag{3}$$

where $\alpha, \beta, \gamma > 0$ are free parameters. The product-of-logs structure satisfies natural boundary conditions ($P \to 0$ with no agents or no time) and saturation ($P \to 100$ with sufficient resources), while capturing diminishing returns across resource axes.

We fit α, β, and γ jointly on all Gemini 3.0 configurations—four subagent counts ($N \in \{1, 2, 4, 8\}$) across 72 hours—yielding $\alpha = 0.972$, $\beta = 4.854$, $\gamma = 2.631$ with $R^2 = 0.98$ (Figure 5).[1] The tight fit indicates that the architecture's gains are smooth and predictable. Notably, the scaling parameter β transfers across LLM backbones: fixing β at the Gemini 3.0 value and recalibrating only α and γ on Gemini 3.1 data from $N \in \{1, 2\}$ accurately predicts the held-out N=8 configuration ($R^2 = 0.92$, RMSE of 3.6 points). This suggests that small-scale calibration runs suffice to predict multi-agent performance under a new backbone. For a more detailed analysis of the scaling laws, cf., Appendix C.

Compute-optimal allocation. Given a fixed budget of $C = N \cdot t$ GPU-hours, optimizing Equation 3 over N at fixed C (Appendix G) yields

$$N^* = [\,\sqrt{\gamma\, C/\beta}\,], \qquad t^* = C/N^*, \qquad P(C) = P(N^*, t^*), \tag{4}$$

where $[x]$ rounds to the nearest natural number ≥ 1. The optimal subagent count grows as $\sqrt{C}$: doubling the budget calls for $\approx 1.4\times$ more subagents rather than simply running longer. This may seem counter-intuitive—in most parallel systems, splitting a fixed budget across more workers degrades efficiency—but because subagents explore independent trajectories without synchronization, additional parallelism increases search diversity, and evolutionary selection retains only the best solutions. This property is specific to our asynchronous evolutionary architecture, not a general claim about multi-agent systems. Figure 6 overlays this *compute frontier* on the empirical data, showing that it closely tracks the upper envelope.

[1] Figures show selected time slices for clarity; the scaling law is fit on $\sim$450,000 observations (Appendix C).

5 Limitations

Data Contamination. Since many Kaggle solutions are publicly available, the underlying LLMs may have encountered them during pre-training. Increased search iterations could raise the probability of "recalling" these solutions rather than generating novel insights, suggesting that evaluation on private benchmarks is necessary to fully isolate research capability. **Preparing the splits.** The HCE protocol requires reformulating each task with additional validation splits, necessitating a degree of human curation during initial task setup; in our experience, configuring the splits for a task takes approximately 15 minutes with an LLM coding assistant, plus roughly 5 minutes of manual verification. This intervention is limited to environment configuration—once initialized, the agent operates autonomously—and is itself automatable given access to the dataset schema. **Compute Specialization.** AIRA_2 is optimized for high-compute regimes and may not be optimal for constrained environments such as short-duration runs or single-GPU setups.

6 Conclusion

We introduce AIRA_2, an agent that addresses three structural bottlenecks constraining AI research agents: compute throughput, evaluation instability, and operator capability. Through an **asynchronous multi-GPU worker pool**, a **Hidden Consistent Evaluation** protocol, and **ReAct agents**, $\text{AIRA}_2^\dagger$ achieves a mean Percentile Rank of **81.5%** at 24 hours and **83.1%** at 72 hours on MLE-bench-30, outperforming the strongest baseline (72.7%). Performance improves consistently with additional compute—without the degradation observed in prior work—and the architecture scales with both compute and model capability, as confirmed by the consistent gains of $\text{AIRA}_2^\dagger$ (Gemini 3.1) over AIRA_2 (Gemini 3.0).

Ablations confirm that each component is necessary: ReAct agents act as an efficiency multiplier for early solution quality; evolutionary selection is required to exploit parallel compute (parallelism without shared state saturates at single-GPU performance); and Hidden Consistent Evaluation is essential for sustained long-horizon improvement, with our analysis revealing that previously reported "overfitting" was driven by evaluation noise rather than data memorization.

Beyond raw performance, the architecture follows a predictable scaling law ($R^2 = 0.98$) that transfers across backbones, enabling principled resource allocation.

Beyond MLE-bench, AIRA_2 exceeds the published human state-of-the-art on 6 out of 20 diverse AIRS-Bench tasks using clean, inductive methodologies—training models from scratch without external data—and approaches SOTA on several others. A systematic audit of all solutions further reveals that autonomous agents readily exploit benchmark shortcuts when available, underscoring the need for robust evaluation environments.

Ultimately, by addressing these fundamental bottlenecks, AIRA_2 moves research agents closer to systems capable of genuine open-ended scientific discovery beyond standard benchmarks.

References

Nikhil Abhyankar, Sanchit Kabra, Saaketh Desai, and Chandan K. Reddy. Accelerating materials design via llm-guided evolutionary search, 2025. URL `https://arxiv.org/abs/2510.22503`.

Anthropic. Claude code overview. `https://code.claude.com/docs/en/overview`, 2025.

Alexis Audran-Reiss, Jordi Armengol-EstapÃŠ, Karen Hambardzumyan, Amar Budhiraja, Martin Josifoski, Edan Toledo, Rishi Hazra, Despoina Magka, Michael Shvartsman, Parth Pathak, et al. What does it take to be a good ai research agent? studying the role of ideation diversity. *arXiv preprint arXiv:2511.15593*, 2025.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305, 2012.

Sai Kiran Botla, Kirubanath Sankar, Abhishek Chopde, and Fardeen Pettiwala. Pi-evolve: Long-horizon evolutionary optimization for autonomous scientific discovery, 2025.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 6989–6997, 2023.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. Mars: Modular agent with reflective search for automated ai research. *arXiv preprint arXiv:2602.02660*, 2026.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Dalpha Team. CobraAgent: Results on MLE-bench, 2026. URL https://dalphakr.github.io/CobraAgent/. GitHub PR: https://github.com/openai/mle-bench/pull/129. Company Website: https://dalpha.so/en.

Shangheng Du, Xiangchao Yan, Dengyang Jiang, Jiakang Yuan, Yusong Hu, Xin Li, Liang He, Bo Zhang, and Lei Bai. Automlgen: Navigating fine-grained optimization for coding agents. *arXiv preprint arXiv:2510.08511*, 2025.

Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017.

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

Johannes Gasteiger, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020.

Google DeepMind. Gemini 3: Our most intelligent AI model, 2025. URL https://deepmind.google/models/gemini/.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomasev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an ai co-scientist, 2025. URL https://arxiv.org/abs/2502.18864.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pages 1–3, 2025.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

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.

Martin Josifoski, Lars Klein, Maxime Peyrard, Nicolas Baldwin, Yifei Li, Saibo Geng, Julian Paul Schnitzler, Yuxing Yao, Jiheng Wei, Debjit Paul, et al. Flows: Building blocks of reasoning and collaborating ai. *arXiv preprint arXiv:2308.01285*, 2023.

Gregory M. Kurtzer, cclerget, Michael Bauer, Ian Kaneshiro, David Trudgian, and David Godlove. hpcng/singularity: Singularity 3.7.3, April 2021. URL https://doi.org/10.5281/zenodo.4667718.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lKEdGCoDNC.

Pat Langley, Herbert A Simon, Gary L Bradshaw, and Jan M Zytkow. Scientific discovery, 1987.

Annan Li, Chufan Wu, Zengle Ge, Yee Hin Chong, Zhinan Hou, Lizhe Cao, Cheng Ju, Jianmin Wu, Huaiming Li, Haobo Zhang, Shenghao Feng, Mo Zhao, Fengzhi Qiu, Rui Yang, Mengmeng Zhang, Wenyi Zhu, Yingying Sun, Quan Sun, Shunhao Yan, Danyu Liu, Dawei Yin, and Dou Shen. The fm agent, 2025. URL https://arxiv.org/abs/2510.26144.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Zexi Liu, Yuzhu Cai, Xinyu Zhu, Yujie Zheng, Runkun Chen, Ying Wen, Yanfeng Wang, Weinan E, and Siheng Chen. Ml-master: Towards ai-for-ai via integration of exploration and reasoning, 2025b. URL https://arxiv.org/abs/2506.16499.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

Alisia Maria Lupidi, Bhavul Gauri, Thomas Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, Jean-Christophe Gagnon-Audet, Chee Hau Leow, Sandra Lefdal, Hossam Mossalam, Abhinav Moudgil, Saba Nazir, Emanuel Tewolde, Isabel Urrego-Gómez, Jordi Armengol-Estapé, Amar Budhiraja, Gaurav Chaurasia, Abhishek Charnalia, Derek Dunfield, Karen Hambardzumyan, Daniel Izcovich, Martin Josifoski, Ishita Mediratta, Kelvin Niu, Parth Pathak, Michael Shvartsman, Edan Toledo, Anton Protopopov, Roberta Raileanu, Alexander H Miller, Tatiana Shavrina, Gabriel Synnaeve, Jakob Nicolaus Foerster, and Yoram Bachrach. AIRS-bench: A suite of tasks for frontier AI research science agents. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=ELg8kRK9Wd.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan O Arik, and Tomas Pfister. MLE-STAR: Machine learning engineering agent via search and targeted refinement. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vS1M06Px6u.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Codex: AI coding agent for ChatGPT. https://chatgpt.com/codex, 2025.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 82–91, 2021.

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

Guillem Simeon and Gianni De Fabritiis. Tensornet: Cartesian tensor representations for efficient learning of molecular potentials. *Advances in Neural Information Processing Systems*, 36:37334–37353, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Gilbert Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 94–101. Elsevier, 1991.

Thesis Labs. Thesis is state-of-the-art on MLE-bench. https://thesislabs.ai/writings/sota-mle-bench, 2025. Accessed: 2026-02-25.

Philipp Thölke and Gianni De Fabritiis. Equivariant transformers for neural network based molecular potentials. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=zNHzqZ9wrRB.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, RISHI HAZRA, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Stenetorp, Nicola Cancedda, Jakob Nicolaus Foerster, and Yoram Bachrach. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=RwfrdKSgCE.

Haorui Wang, Marta Skreta, Cher Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=awWiNvQwf3.

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. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=OJd3ayDDoF.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jiaxin Wen, Liang Qiu, Joe Benton, Jan Hendrik Kirchner, and Jan Leike. Automated weak-to-strong researcher, 2026. URL https://alignment.anthropic.com/2026/automated-w2s-researcher/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

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

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Cunxi Yu, Rongjian Liang, Chia-Tung Ho, and Haoxing Ren. Autonomous code evolution meets np-completeness. *arXiv preprint arXiv:2509.07367*, 2025.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Ruiyi Zhang, Peijia Qin, Qi Cao, Li Zhang, and Pengtao Xie. Aibuildai: An ai agent that automatically builds ai models, 2026.

Xinyu Zhu, Yuzhu Cai, Zexi Liu, Bingyang Zheng, Cheng Wang, Rui Ye, Jiaao Chen, Hanrui Wang, Wei-Chen Wang, Yuzhi Zhang, et al. Toward ultra-long-horizon agentic science: Cognitive accumulation for machine learning engineering. *arXiv preprint arXiv:2601.10402*, 2026.

A Implementation Details

This section provides additional implementation details for the AIRA_2 system described in Section 3.2.

Stateful Interaction. Unlike previous systems such as AIDE [Jiang et al., 2025] and AIRA-dojo, our Bash command and Jupyter kernel execution tools are stateful. This allows agents to maintain context across multiple turns, enabling a more interactive and iterative debugging process. Tool outputs also include execution duration, allowing agents to monitor their own efficiency.

Resource Allocation. We enforce a strict 1:1 worker-to-GPU mapping using NVIDIA H200 GPUs (141GB VRAM). Each worker is allocated 12 logical CPU cores and a dedicated 120GB of system RAM, providing sufficient compute for training large models, running hyperparameter sweeps, or building deep ensembles without resource contention.

Evolutionary Search. The population grows without a size limit; all candidate solutions are retained throughout the search. Search is initialized from an empty program (no seed solution or template). For mutation, the agent receives the parent program's source code, its stdout/stderr output, and the wall-clock time taken to execute it. For crossover, the agent receives two parent programs with the same metadata and is instructed to produce a new program inspired by both. Each search instance targets a single task.

LLM Configuration. All LLM calls use the default generation parameters of the respective backbone (Gemini 3.0 or 3.1 Pro Preview) without modification. When agent trajectories exceed the model's context window, they are truncated to fit.

Data Management & Limits. The program database resides in-memory for fast access, with large artifacts automatically offloaded to hard disk. Subagents communicate exclusively through this central database, contributing ideas and code asynchronously. To optimize resource utilization, evaluation is performed in the foreground without a separate job queue, running in an identical container with a dedicated GPU. The search process continues until a global hard limit is reached, with individual code executions capped at a 9-hour hard time limit to prevent stalled processes.

B Baseline Scaffold Comparison

To provide a reference point with an established, widely used agent, we compare Mini-SWE-Agent† [Yang et al., 2024b] with $\text{AIRA}_2^\dagger$ (1 GPU) and $\text{AIRA}_2^\dagger$ with 8 subagents. To make use of HCE and the evaluation feedback, Mini-SWE-Agent† is run within our pipeline in place of the ReAct subagent of $\text{AIRA}_2^\dagger$ (1 GPU), so the comparison between the two 1-GPU configurations isolates the choice of agent. All three use `Gemini 3.1 Pro` under the same HCE protocol, execution environment, and hardware, and are compared at a matched total GPU-hour budget (Table 2). We report task-weighted Percentile Rank on MLE-bench-30, taking the running best solution selected on the held-out selection split and scoring it on the test set.

The ordering is budget-dependent. Mini-SWE-Agent† leads through 3 GPU-hours, $\text{AIRA}_2^\dagger$ (1 GPU) leads from 6 to 12 GPU-hours, and $\text{AIRA}_2^\dagger$ with 8 subagents starts slowest—owing to the initial cost of building a diverse population—but leads at every budget from 24 GPU-hours onward (71.7, matching $\text{AIRA}_2^\dagger$ at 3 hours wall-clock in Table 1). This lead persists with further compute: at 84 GPU-hours, the 8-subagent configuration is 4.7 points above $\text{AIRA}_2^\dagger$ (1 GPU) and 6.2 points above Mini-SWE-Agent†. Moreover, the two single-GPU configurations are flattening while the 8-subagent configuration is not: between 72 and 84 GPU-hours, Mini-SWE-Agent† gains 0.5 points and $\text{AIRA}_2^\dagger$ (1 GPU) gains 0.9, against 2.0 for the 8-subagent configuration. In the long run, the two agents perform similarly in the single-worker setting (73.1 vs. 74.6 at 84 GPU-hours), indicating that the gains of $\text{AIRA}_2^\dagger$ stem from scaling the number of parallel subagents rather than from the specific choice of agent. This comparison is conservative relative to Table 1: for the 8-subagent configuration, 84 GPU-hours corresponds to only 10.5 hours of wall-clock time, well before it reaches 81.5 at 24 hours or 83.1 at 72 hours.

Table 2: **Baseline scaffold comparison.** Percentile Rank on MLE-bench-30 for Mini-SWE-Agent† and AIRA$_2^\dagger$ using `Gemini 3.1 Pro` under a matched GPU-hour budget; all configurations run within our pipeline under the HCE protocol, with the same execution environment and hardware, for a like-for-like comparison. The best entry at each budget is shown in bold.

GPU-hours	Mini-SWE-Agent† (1 GPU)	AIRA$_2^\dagger$ (1 GPU)	AIRA$_2^\dagger$ (8 subagents)
1	**37.0**	9.5	0.0
2	**47.0**	22.6	4.1
3	**49.6**	39.0	13.1
6	53.1	**56.8**	36.9
9	56.4	**59.9**	51.4
12	58.7	**64.9**	59.4
24	64.4	69.4	**71.7**
30	65.3	70.3	**73.4**
36	65.8	71.9	**74.2**
42	67.7	72.7	**75.1**
48	68.5	72.8	**75.4**
60	71.0	73.5	**77.0**
72	72.6	73.7	**77.3**
84	73.1	74.6	**79.3**

C Extended Scaling Law Analysis

C.1 Fitting Procedure

Data construction. For each subagent configuration, we compute the task-averaged mean percentile rank as a running best curve over time across all 30 MLE-bench tasks. These per-configuration curves are aligned onto a shared time grid formed by the union of all event timestamps, yielding approximately 150,000 unique time points for Gemini 3.0 Pro and 59,000 for Gemini 3.1 Pro, with a median resolution of under 10 seconds. Each (N, t) pair where the mean score is positive contributes one data point, resulting in approximately 450,000 observations for Gemini 3.0 Pro (across $N \in \{1, 2, 4, 8\}$) and 165,000 for Gemini 3.1 Pro ($N \in \{1, 2, 8\}$), all capped at $t = 72$ hours. We fit Equation 3 jointly on the full set of observations by minimizing squared residuals using Levenberg–Marquardt optimization. Figures in the main text show selected time slices for clarity.

C.2 Cross-Model Transferability

A practical question is whether the scaling law can predict performance under unseen configurations, such as a new LLM backbone, with minimal calibration. To test this, we hold out the $N = 8$ subagent configuration from the Gemini 3.1 Pro data and attempt to predict it from $N \in \{1, 2\}$ alone.

With only two subagent counts, the three parameters α, β, and γ are poorly constrained: β governs how performance scales with the number of agents, but only two nearby values of N provide little leverage to estimate it. We posit that β reflects the *multi-agent architecture*—how tasks are decomposed, how subagents communicate, and how solutions are selected—rather than properties of the backbone LLM itself. In that case, β should transfer across backbones, while α (saturation rate) and γ (per-agent time scaling) may differ.

We therefore fix $\beta = 4.854$ (the Gemini 3.0 Pro value) and fit only α and γ on the Gemini 3.1 Pro data restricted to $N \in \{1, 2\}$. The recalibrated law yields $\alpha = 2.673$, $\gamma = 0.481$, and achieves $R^2 = 0.92$ on the full dataset including the held-out $N = 8$ configuration, with a held-out RMSE of 3.6 percentile-rank points (Figure 5). The lower γ relative to Gemini 3.0 Pro (0.481 vs. 2.631) reflects a difference in time dynamics: Gemini 3.1 Pro improves rapidly in the first few hours then plateaus, whereas Gemini 3.0 Pro continues to gain more gradually over the full 72-hour horizon. Despite this difference, the shared β produces accurate extrapolation from 2 to 8 subagents, supporting the hypothesis that the agent-count scaling parameter reflects the multi-agent architecture rather than backbone-specific characteristics. This suggests that a small-scale calibration run—as few as two

subagent counts—may suffice to predict multi-agent performance under a new backbone, provided the architectural scaling parameter is carried over from a prior fit.

D Evaluation Failure: A Concrete Example

To illustrate how implementation bugs can silently corrupt the search signal (Section 2), we present a real example from an AI agent solving the LMSYS Chatbot Arena competition on MLE-bench. The agent's solution reported a *perfect* cross-validation log-loss of 0.0, which the search process then treated as the best candidate—despite the underlying model being unremarkable.

The bug. The competition requires predicting which of three outcomes occurred (`winner_model_a`, `winner_model_b`, or `winner_tie`). The agent converts multi-hot target columns to single labels via `idxmax(axis=1)`, which returns *column name strings*, not integer indices:

```
target_cols = ["winner_model_a", "winner_model_b", "winner_tie"]
train_labels = train_df[target_cols].idxmax(axis=1).values
# Agent comment: "0, 1, 2" -- actually returns strings:
# ['winner_model_b', 'winner_model_a', 'winner_model_b', ...]
```

When computing the validation metric, the agent passes `labels=[0, 1, 2]` (integers), creating a type mismatch:

```
loss = log_loss(train_labels[val_idx], val_pred, labels=[0, 1, 2])
# Returns 0.0 regardless of predictions
```

Because `scikit-learn`'s `log_loss` cannot match the string ground-truth labels (`'winner_model_a'`, etc.) with the integer label list (`[0, 1, 2]`), it silently returns 0.0 for *any* set of predictions:

```
>>> train_labels[:3]
array(['winner_model_b', 'winner_model_a', 'winner_model_b'], dtype=object)
>>> val_pred = [[0.3, 0.3, 0.4]] * 3  # arbitrary predictions
>>> log_loss(train_labels[:3], val_pred, labels=[0, 1, 2])
0.0
```

Why this matters for search. This bug is particularly insidious in the context of search-based agents. A greedy or evolutionary search process that relies on self-reported validation scores would select this solution as the global optimum, halting further exploration. The solution would persist as the "best" candidate indefinitely, while its actual test performance would be no better than chance. This example concretely demonstrates why decoupling the search signal from agent-controlled evaluation—as in our Hidden Consistent Evaluation protocol (Section 3.3)—is essential for reliable long-horizon search.

E MLE-bench

Table 3: Per-task Percentile Rank scores for AIRA_2 at different time cutoffs. Values shown as score with 95% CI below.

Task	3h	12h	24h	72h
	70.30	87.66	97.31	97.35
aptos2019-blindness-detection	51.16, 93.31	64.79, 99.76	93.75, 99.76	96.41, 98.16
	66.67	100.00	100.00	100.00
billion-word-imputation	8.05, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	19.72	28.15	57.67	69.07
bms-molecular-translation	19.45, 20.25	19.45, 34.10	32.95, 80.66	60.64, 85.93
	58.85	87.38	84.73	89.15
cassava-leaf-disease-classification	39.49, 88.97	82.90, 90.26	77.59, 90.26	86.36, 92.13
	30.72	49.17	63.12	78.13
champs-scalar-coupling	9.02, 45.80	33.02, 64.10	41.09, 94.34	60.56, 98.79
	98.16	99.68	100.00	100.00
freesound-audio-tagging-2019	97.60, 98.56	99.04, 100.00	100.00, 100.00	100.00, 100.00
	27.62	29.03	29.21	29.60
h-and-m-personalized-fashion-recommendations	27.06, 28.65	28.75, 29.53	28.95, 29.53	29.19, 30.00
	30.30	30.57	30.28	31.76
hms-harmful-brain-activity-classification	29.50, 31.09	29.25, 31.89	29.25, 31.31	31.63, 31.89
	85.51	89.13	89.86	94.57
hotel-id-2021-fgvc8	82.61, 89.13	89.13, 89.13	89.13, 91.30	92.39, 95.65
	89.27	93.27	93.91	98.92
hubmap-kidney-segmentation	87.32, 90.49	89.49, 97.58	89.49, 99.50	97.58, 99.58
	38.60	52.63	71.93	85.61
imet-2020-fgvc7	37.89, 40.00	46.32, 55.79	47.37, 84.21	83.16, 88.42
	8.05	8.05	8.06	8.37
jigsaw-unintended-bias-in-toxicity-classification	7.86, 8.39	7.90, 8.36	7.90, 8.36	7.94, 8.85
	86.69	94.43	95.34	97.50
kuzushiji-recognition	69.97, 97.61	84.30, 100.00	86.01, 100.00	92.49, 100.00
	96.67	97.92	98.33	98.75
mlsp-2013-birds	95.00, 98.75	96.25, 98.75	97.50, 98.75	97.50, 100.00
	62.96	70.37	70.37	81.48
multi-modal-gesture-recognition	38.89, 100.00	44.44, 100.00	44.44, 100.00	55.56, 100.00
	37.47	38.79	38.79	37.04
new-york-city-taxi-fare-prediction	35.58, 40.09	35.58, 40.50	35.58, 40.50	32.95, 40.30
	42.49	71.07	71.92	82.73
nfl-player-contact-detection	23.22, 79.23	25.03, 94.46	26.30, 94.99	58.09, 95.63
	99.66	99.58	99.66	99.77
nomad2018-predict-transparent-conductors	99.54, 99.89	99.54, 99.66	99.54, 99.89	99.66, 100.00
	13.85	10.89	11.02	12.50
osic-pulmonary-fibrosis-progression	12.40, 15.70	10.07, 11.50	10.07, 11.88	11.88, 13.22
	54.37	56.41	63.37	88.62
petfinder-pawpularity-score	52.78, 55.75	54.11, 59.88	51.12, 83.74	68.45, 99.72
	100.00	100.00	100.00	100.00
plant-pathology-2021-fgvc8	100.00, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	4.89	5.00	5.00	9.02
smartphone-decimeter-2022	4.89, 4.89	4.89, 5.06	4.89, 5.06	4.89, 16.75
	97.29	98.04	97.90	98.39
spooky-author-identification	96.45, 98.15	97.99, 98.07	97.26, 98.31	98.31, 98.47
	100.00	100.00	100.00	100.00
stanford-covid-vaccine	100.00, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	48.91	94.30	97.27	97.73
tensorflow2-question-answering	48.66, 49.15	88.97, 97.89	96.51, 97.89	96.51, 98.54
	73.34	96.28	95.92	93.73
tweet-sentiment-extraction	60.39, 83.12	95.95, 96.49	93.57, 98.25	85.86, 98.16
	96.52	99.51	99.63	99.91
us-patent-phrase-to-phrase-matching	91.53, 99.15	99.21, 99.89	99.42, 100.00	99.79, 100.00
	18.06	22.54	26.18	33.35
uw-madison-gi-tract-image-segmentation	4.91, 43.83	5.30, 56.43	5.04, 67.61	5.04, 89.08
	48.78	55.40	55.93	61.78
ventilator-pressure-prediction	47.96, 49.39	50.02, 63.52	50.38, 63.52	60.83, 63.52
	92.11	98.55	98.55	99.31
whale-categorization-playground	86.17, 96.59	97.35, 99.24	97.35, 99.24	98.30, 99.81

Table 4: Per-task test percentile scores for $\text{AIRA}_2^\dagger$ at different time cutoffs. Values shown as score with 95% CI below.

Task	3h	12h	24h	72h
	92.59	95.75	95.75	96.60
aptos2019-blindness-detection	83.67, 98.22	88.70, 99.39	88.70, 99.39	93.03, 99.25
	97.70	100.00	100.00	100.00
billion-word-imputation	93.10, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	16.40	80.97	86.08	93.63
bms-molecular-translation	0.00, 49.20	73.68, 85.13	84.44, 87.41	92.56, 95.08
	87.98	88.40	96.60	99.06
cassava-leaf-disease-classification	82.90, 94.69	82.90, 95.95	90.26, 99.95	97.26, 99.97
	95.05	99.06	99.27	99.53
champs-scalar-coupling	92.70, 96.38	98.69, 99.31	99.01, 99.45	99.49, 99.56
	95.84	99.36	100.00	100.00
freesound-audio-tagging-2019	93.05, 97.84	98.08, 100.00	100.00, 100.00	100.00, 100.00
	29.03	30.82	33.53	34.07
h-and-m-personalized-fashion-recommendations	28.95, 29.19	29.22, 31.83	29.46, 36.40	29.22, 36.57
	24.21	27.90	28.48	28.21
hms-harmful-brain-activity-classification	21.04, 26.32	24.37, 31.31	26.68, 30.73	26.46, 31.49
	91.30	95.29	97.10	98.55
hotel-id-2021-fgvc8	90.22, 92.39	93.48, 96.74	96.74, 97.83	96.74, 100.00
	89.27	95.25	96.19	98.25
hubmap-kidney-segmentation	85.15, 92.16	89.41, 98.42	92.58, 99.67	97.50, 99.67
	64.21	85.26	87.72	92.63
imet-2020-fgvc7	53.68, 83.16	85.26, 85.26	85.26, 90.53	91.58, 94.74
	7.81	8.20	8.34	8.66
jigsaw-unintended-bias-in-toxicity-classification	7.75, 7.90	7.90, 8.70	7.98, 8.70	8.36, 9.23
	99.54	100.00	100.00	100.00
kuzushiji-recognition	98.63, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	95.00	96.67	99.17	99.17
mlsp-2013-birds	90.00, 98.75	93.75, 98.75	98.75, 100.00	98.75, 100.00
	100.00	100.00	100.00	100.00
multi-modal-gesture-recognition	100.00, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	39.71	38.54	36.97	38.63
new-york-city-taxi-fare-prediction	36.05, 42.45	36.05, 41.78	36.05, 37.80	36.66, 40.90
	92.08	95.03	95.56	96.77
nfl-player-contact-detection	88.29, 94.46	94.14, 95.95	93.82, 96.81	96.06, 97.12
	100.00	100.00	100.00	100.00
nomad2018-predict-transparent-conductors	100.00, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	12.56	12.53	12.02	12.34
osic-pulmonary-fibrosis-progression	12.31, 12.88	11.12, 14.17	11.12, 13.07	12.17, 12.45
	75.26	94.19	98.13	98.70
petfinder-pawpularity-score	58.75, 99.46	93.13, 96.21	96.10, 100.00	96.10, 100.00
	100.00	100.00	100.00	100.00
plant-pathology-2021-fgvc8	100.00, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	8.96	40.78	36.77	36.59
smartphone-decimeter-2022	4.89, 16.75	4.89, 100.00	4.89, 100.00	4.89, 100.00
	98.25	98.74	98.93	99.03
spooky-author-identification	98.15, 98.31	98.55, 98.95	98.87, 98.95	98.95, 99.11
	100.00	100.00	100.00	100.00
stanford-covid-vaccine	100.00, 100.00	100.00, 100.00	100.00, 100.00	100.00, 100.00
	48.91	80.35	95.59	98.54
tensorflow2-question-answering	48.66, 49.07	49.47, 97.40	92.62, 97.97	97.89, 99.43
	88.13	97.93	98.53	98.25
tweet-sentiment-extraction	79.00, 96.31	96.90, 98.52	98.20, 99.10	97.44, 99.10
	99.51	99.93	100.00	100.00
us-patent-phrase-to-phrase-matching	99.42, 99.68	99.84, 100.00	100.00, 100.00	100.00, 100.00
	61.39	81.02	87.90	93.95
uw-madison-gi-tract-image-segmentation	57.27, 68.07	59.21, 93.21	80.38, 93.21	89.40, 97.29
	46.17	53.94	58.74	70.06
ventilator-pressure-prediction	44.62, 47.20	52.34, 55.76	55.57, 61.18	60.48, 88.10
	94.63	96.84	98.36	99.56
whale-categorization-playground	91.86, 96.59	96.21, 97.35	97.35, 99.43	99.43, 99.81

F AIRS-Bench

Table 5: Per-task results on AIRS-Bench. Each task group shows AIRA_2 followed by the two best-performing baselines (selected by best seed). Normalised scores are scaled so that 0 = worst and 100 = SOTA. ∞ denotes normalised scores whereby 1 seed achieved the optimal score. This is usually due to unfair solutions. See Section H.2. We show the best baseline per task selected by the best seed score since we care about SOTA beating solutions. **Bold** indicates the best method per task (by mean normalised score).

Task	Method	SOTA	Best	Mean	Norm. Best	Norm. Mean	Seeds	Beats SOTA
Code Gen.	**AIRA_2**	0.1870	**0.2810**	**0.2309**	**159.4**	**127.3**	**3**	**3**
	$AIRA_{\text{greedy}}$ GPT5		0.0050	0.0020	2.4	1.0	10	0
Code Retrieval	**AIRA_2**	0.6113	**0.5772**	**0.5037**	**91.1**	**74.8**	**3**	**0**
	$AIRA_{\text{greedy}}$ GPT5		0.4485	0.1968	63.0	23.2	10	0
Coref. (WSC)	**AIRA_2**	0.9620	**1.0000**	**0.9952**	∞	∞	**4**	**4**
	$AIRA_{\text{greedy}}$ GPT5		0.9423	0.7577	85.2	34.2	10	0
Coref. (Wino.)	**AIRA_2**	0.8540	**0.9132**	**0.9092**	**130.4**	**127.9**	**4**	**4**
	$AIRA_{\text{greedy}}$ GPT5		0.9148	0.6938	131.5	56.6	10	4
Mol. Prop. (C_v)	**AIRA_2**	0.0210	**0.0193**	**0.0199**	**100.8**	**100.5**	**4**	**4**
	$AIRA_{\text{greedy}}$ GPT5		0.0306	0.0557	96.4	90.8	10	0
Mol. Prop. (G)	**AIRA_2**	7.5300	**5.5464**	**5.6387**	**101.1**	**101.0**	**4**	**4**
	$AIRA_{\text{greedy}}$ o3		17.6286	3.39×10^4	97.0	70.0	10	0
Graph Reg.	**AIRA_2**	0.0170	**0.0226**	**0.0227**	**93.9**	**93.7**	**4**	**0**
	$AIRA_{\text{greedy}}$ o3		0.0408	0.0984	81.1	62.0	10	0
Math QA	**AIRA_2**	0.9420	**0.9367**	**0.9267**	**96.9**	**92.2**	**4**	**0**
	$AIRA_{\text{greedy}}$ o3		0.6500	0.4253	36.9	19.5	10	0
QA (DuoRC)	**AIRA_2**	0.4648	**0.3925**	**0.3842**	**79.7**	**77.6**	**4**	**0**
	$AIRA_{\text{oneshot}}$ OSS-120B		0.3262	0.2782	63.2	52.2	20	0
QA (ELI5)	**AIRA_2**	0.2690	**0.2523**	**0.2483**	**92.7**	**91.0**	**4**	**0**
	$AIRA_{\text{greedy}}$ GPT5		0.2354	0.1881	85.6	66.2	10	0
QA (FinQA)	**AIRA_2**	0.7803	**1.0000**	**0.6713**	∞	∞	**4**	**1**
	$AIRA_{\text{greedy}}$ GPT5		0.0924	0.0340	6.4	2.3	10	0
Mol. Prop. (R^2)	**AIRA_2**	0.0330	**0.0221**	**0.0241**	**103.6**	**102.8**	**4**	**4**
	$AIRA_{\text{greedy}}$ GPT5		0.1550	0.5619	86.2	74.7	10	0
Reading Comp.	**AIRA_2**	0.8580	**0.8437**	**0.8316**	**95.1**	**91.4**	**4**	**0**
	$AIRA_{\text{greedy}}$ GPT5		0.8645	0.5187	102.4	37.5	10	2
Sentiment	**AIRA_2**	0.7780	**0.7311**	**0.7304**	**85.1**	**84.9**	**4**	**0**
	$AIRA_{\text{greedy}}$ GPT5		0.7165	0.6671	80.9	68.4	10	0
Text Cls.	**AIRA_2**	0.9050	**0.9333**	**0.9298**	**116.1**	**113.8**	**4**	**4**
	$AIRA_{\text{greedy}}$ o3		0.9327	0.9012	115.7	98.2	10	9
Text Sim.	**AIRA_2**	0.8540	**0.9083**	**0.9070**	**124.1**	**123.3**	**4**	**4**
	$AIRA_{\text{greedy}}$ GPT5		0.9078	0.8737	123.8	107.5	10	8
TS (Traffic)	**AIRA_2**	0.6220	4.72×10^{11}	4.72×10^{11}	**18.8**	**12.5**	**3**	**0**
	$AIRA_{\text{greedy}}$ OSS-120B		4.28×10^{11}	5.82×10^{12}	19.1	11.4	10	0
TS (Rideshare)	**AIRA_2**	1.1850	**1.0661**	**1.1528**	**105.6**	**101.5**	**3**	**1**
	$AIRA_{\text{greedy}}$ OSS-20B		1.1741	1.2884	100.5	95.6	10	2
TS (Solar)	**AIRA_2**	576.35	8.06×10^2	1.06×10^3	**83.7**	**72.0**	**3**	**0**
	$AIRA_{\text{greedy}}$ GPT5		7.49×10^2	1.29×10^3	87.3	61.1	10	0
Mol. Prop. (U_0)	**AIRA_2**	5.83	**4.4481**	**6.8336**	**101.0**	**99.9**	**4**	**3**
	$AIRA_{\text{greedy}}$ GPT5		11.9779	77.3832	97.5	90.9	10	0

G Compute Optimal Resource Allocation

Since $P = 100 \cdot g/(g+1)$ is strictly monotonically increasing in g, and $g = \alpha \cdot h$ with $\alpha > 0$, maximizing P over (N, t) subject to a fixed compute budget $C = N \cdot t$ reduces to maximizing

$$h(N, t) = \log(\beta\, N + 1) \cdot \log(\gamma\, t + 1). \tag{5}$$

Theorem 1. *Given the compute budget constraint $C = N \cdot t$ with $C, \beta, \gamma > 0$, the function $h(N, t) = \log(\beta\, N + 1) \cdot \log(\gamma\, t + 1)$ achieves its unique global maximum at:*

$$N^* = \sqrt{\frac{\gamma C}{\beta}}, \qquad t^* = \sqrt{\frac{\beta C}{\gamma}}. \tag{6}$$

Proof. Substituting $t = \frac{C}{N}$ into the objective, we obtain a single-variable function $h(N) = \log(\beta\, N + 1) \cdot \log\left(\frac{\gamma C}{N} + 1\right)$. To find the stationary points, we introduce auxiliary variables $x \triangleq \beta N$ and $y \triangleq \frac{\gamma C}{N}$. Note that $x \cdot y = \beta\gamma C$, which is a constant independent of N.

Setting the derivative of $h(N)$ with respect to N to zero yields:

$$\frac{dh}{dN} = \frac{\beta}{1 + \beta N} \log\left(1 + \frac{\gamma C}{N}\right) - \frac{\gamma C / N^2}{1 + \frac{\gamma C}{N}} \log(1 + \beta N) = 0. \tag{7}$$

Substituting x, y, and the identities $\beta = \frac{x}{N}$ and $\frac{\gamma C}{N^2} = \frac{y}{N}$ into the stationarity condition, we have:

$$\frac{x/N}{1 + x} \log(1 + y) - \frac{y/N}{1 + y} \log(1 + x) = 0. \tag{8}$$

Multiplying by N and separating the variables to opposite sides of the equation gives:

$$\frac{1 + x}{x} \log(1 + x) = \frac{1 + y}{y} \log(1 + y). \tag{9}$$

Observe that this equation is of the form $f(x) = f(y)$, where $f(z) \triangleq \frac{1+z}{z} \log(1 + z)$. To determine the injectivity of $f(z)$ for $z > 0$, we evaluate its derivative:

$$f'(z) = \frac{z - \log(1 + z)}{z^2}. \tag{10}$$

By the standard logarithmic inequality $z > \log(1 + z)$ for all $z > 0$, it follows that $f'(z) > 0$. Therefore, $f(z)$ is strictly monotonically increasing, implying that $f(x) = f(y)$ if and only if $x = y$.

Equating $x = y$ and substituting their original definitions back yields:

$$\begin{aligned} \beta N &= \frac{\gamma C}{N} \\ N^* &= \sqrt{\frac{\gamma C}{\beta}}. \end{aligned} \tag{11}$$

Applying the constraint $t = \frac{C}{N}$ gives the corresponding optimal time allocation $t^* = \sqrt{\frac{\beta C}{\gamma}}$, concluding the proof. □

Remark (Optimal Performance Trajectory). At the optimum, both logarithmic arguments equalize to $\sqrt{\beta\gamma C} + 1$, so $h^* = \left[\log\left(\sqrt{\beta\gamma C} + 1\right)\right]^2$ and $g^* = \alpha \cdot h^*$. Substituting into $P = 100 \cdot g/(g+1)$ gives the maximum achievable performance as a function of the total compute budget C:

$$P(C) = 100 \cdot \frac{\alpha \left[\log\left(\sqrt{\beta\gamma C} + 1\right)\right]^2}{\alpha \left[\log\left(\sqrt{\beta\gamma C} + 1\right)\right]^2 + 1}. \tag{12}$$

Remark (Integer Constraint). Since N must be a positive integer in practice, we round N^* to the nearest integer, $N^* \leftarrow [N^*]$, and set $t^* = C/N^*$.

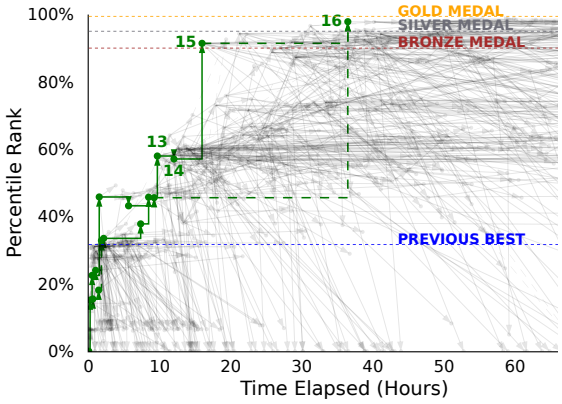
H Case Studies

H.1 Case Study 1: Eureka Moments in MLE-bench

Overview & Takeaways: To understand the qualitative behaviours driving AIRA$_2$'s quantitative success, we analyse its trajectory on specific high-complexity tasks. We find that AIRA$_2$ appears to exhibit reasoning capabilities—specifically the ability to distinguish between poor methodology and poor execution (e.g., underfitting)—allowing it to recover from local minima where greedy agents would likely fail.

H.1.1 Deep Dive: Predicting Molecular Properties

We highlight the agent's performance on `champs-scalar-coupling`, a task predicting magnetic interactions between atom pairs. As illustrated in Figure 7, AIRA$_2$ demonstrates a "eureka" moment.



AGENT'S THOUGHT AND ACTION TRAJECTORY SUMMARY

13, Mutation: The previous approach with SchNet significantly improved the score, and ran very quickly. Due to the competition's evaluation metric-mean log MAE calculated per coupling type with varied value ranges-we will decouple and train separate models for each type.

14, Mutation: The previous solution achieved a higher score by implementing 8 separate SchNet models. Next, we will be using Mulliken charges as an auxiliary prediction task during training.

15, Mutation: The parent's *score was lower* than the grandparent's, with training taking only 15 minutes-well within the 9-hour budget-*yet the loss was converging well*. This suggests that the model *likely underfit* or stopped prematurely, failing to fully benefit from the auxiliary task. My next step is to *train a larger model for a longer duration*. → 🥉

16, Crossover: The parent A's approach can be further scaled up to improve the performance. I will combine it with parent B's efficient data preprocessing strategy. → 🥈

Figure 7: **Example of typical behaviour observed in AIRA$_2$ during the `champs-scalar-coupling` task.** The left side displays the chronological performance of each solution in the database (solid lines for mutation, dashed for crossover). The right-hand panel presents a concise summary of the agent's thought process for the nodes and trajectory annotated in green, highlighting the "eureka" moment where the agent identifies underfitting and subsequently scales the model to achieve medal-winning performance. Horizontal dashed lines indicate the medal thresholds and display the previous best attempt [Thesis Labs, 2025] on the task among all agents.

The shown trajectory begins with the agent testing `SchNet` [Schütt et al., 2017], which performed well. Building on this, the agent attempted to introduce an auxiliary prediction task (Mulliken charges) at Step 14, leveraging supplementary competition data to further improve results. As shown in the trace, this caused the validation score to drop. A standard ReAct loop or greedy algorithm might reject this change and revert. However, AIRA$_2$ investigates the execution logs, noting that the training only utilized 15 minutes of the available 9-hour budget and that the loss was still converging, correctly identifying the root cause as *underfitting* — recognizing that the auxiliary task was effective despite the lack of immediate improvement — rather than a fundamental flaw in the idea.

Consequently, at Step 15, the agent commits to scaling the model size and extending the training duration significantly. This reasoning leads directly to a dramatic performance boost, securing a Bronze medal and surpassing all previous agents. Following this improvement, AIRA$_2$ further refined the approach by adopting an effective preprocessing technique via crossover from a much weaker solution. This combination elevated the solution to a Silver and ultimately a Gold medal. Notably, no other reported agent (FM-Agent 2.0, MARS, etc.) achieved a medal on this task.

Breaking the Ceiling on Stalled Tasks. To illustrate that this exceptional performance is not an isolated event, we briefly outline two other complex tasks where AIRA$_2$ uniquely surpassed the medal threshold despite the failure of all prior methods.

On `billion-word-imputation`, a challenging NLP task where baselines failed to make significant progress, AIRA$_2$ achieved a 100% Percentile Rank by decomposing the problem into two learned

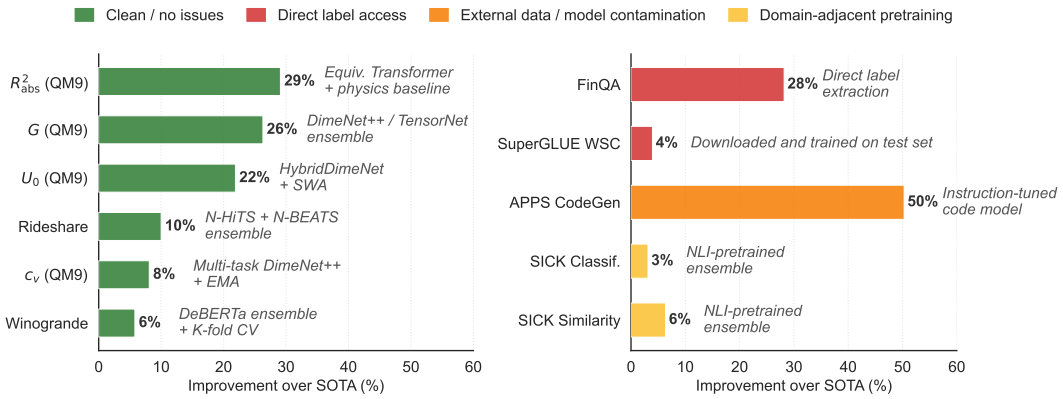


Figure 8: AIRS-Bench SOTA-beating tasks split by integrity status. **Left:** Six tasks where AIRA$_2$ exceeded SOTA using clean, inductive methodologies (models trained from scratch, no external data). Italic annotations indicate key techniques. **Right:** Five tasks where the SOTA-beating solutions were flagged with integrity concerns, color-coded by severity: direct test label access (red), external data or model contamination (orange), and domain-adjacent pretraining advantage (amber). Bars show percentage improvement over published SOTA.

sub-tasks: a RoBERTa-large token classifier trained on 8M synthetically constructed gapped sentences to *detect* the missing-word position, followed by a separately fine-tuned RoBERTa-large masked language model to *fill* the gap.

On the fine-grained visual classification task `imet-2020-fgvc7` (3,474 attribute classes), AIRA$_2$ achieved a 91% Percentile Rank—the highest among all evaluated agents—by ensembling an EVA-02 Large [Fang et al., 2024] and a ConvNeXt Large CLIP [Liu et al., 2022] model with asymmetric loss [Ridnik et al., 2021], layer-wise learning rate decay, and grid-searched ensemble weights and classification thresholds.

H.2 Case Study 2: Pushing the Frontier with AIRS-Bench

Overview & Takeaways. To examine how AIRA$_2$ behaves beyond the Kaggle-style competitions of MLE-bench, we evaluated it on AIRS-Bench [Lupidi et al., 2026], a suite of 20 diverse machine learning tasks spanning molecular property prediction, graph regression, time series forecasting, natural language understanding, code generation, and mathematical reasoning. Unlike MLE-bench, which is drawn from structured competitions, AIRS-Bench is curated from recent research problems and aims to capture the breadth and open-endedness of modern AI research. In this case study, we used the same system configuration as our MLE-bench experiments (8× H200 GPUs, ∼72 hours) with 3 seeds per task.

Across the 20 tasks, AIRA$_2$ exceeded the AIRS-bench recorded state-of-the-art (SOTA) results on 11 of them. However, a systematic manual audit of all solution code revealed a clear domain split (Figure 8): **6 of these 11 successes used clean, inductive methodologies with no detected integrity issues,** while the remaining 5 relied on data contamination, benchmark shortcuts, or domain-adjacent model selection that conferred unfair advantages. The clean successes were concentrated in tasks where models must be trained from scratch—molecular property prediction and time series forecasting—while every SOTA-beating NLP or code generation task exhibited at least one potential integrity concern. This asymmetry highlights two crucial takeaways for the field. First, quantitative results from autonomous agents require auditing, as aggregate scores cannot distinguish genuine methodological improvement from data contamination. Second, benchmarks must anticipate adversarial optimization; tasks with freely available test data or well-known public datasets are highly vulnerable to autonomous exploitation, even without explicit instructions to cheat. The ongoing challenge is designing evaluation environments where rigorous problem-solving—rather than exploitation—is the path of least resistance.

H.2.1 Deep Dive: Methodological Improvements.

The most compelling legitimate results emerged from tasks where models were trained entirely from scratch, with no pretrained model weights or external data downloads. The QM9 molecular property prediction suite was the standout, with AIRA_2 exceeding SOTA on all four tasks:

- **Electronic Spatial Extent (R^2_{abs}):** Achieved a **29% improvement** (MAE of 0.023 vs. 0.033 Bohr^2) using a TorchMD Equivariant Transformer [Thölke and Fabritiis, 2022] ensemble with a physics-informed RidgeCV baseline that incorporated charge-weighted spatial moments ($\sum Z_i r_i^2$) and rotational constants computed directly from atomic coordinates.
- **Free Energy (G):** Achieved a **26% improvement** (MAE of 5.55 vs. 7.53 meV) by constructing a heterogeneous ensemble combining DimeNet++ [Gasteiger et al., 2020] and TensorNet [Simeon and De Fabritiis, 2023] architectures, with a linear regression composition baseline for residual learning.
- **Internal Energy (U_0):** Achieved a **22% improvement** through a hybrid architecture (DimeNet++ [Gasteiger et al., 2020] augmented with an MLP branch processing log-transformed rotational constants), trained with Stochastic Weight Averaging [Izmailov et al., 2018] (SWA) and Huber loss across a 5-model bagging ensemble.
- **Heat Capacity (c_v):** Achieved an **8% improvement** using a 5-seed DimeNet++ [Gasteiger et al., 2020] ensemble with Exponential Moving Average (EMA), jointly predicting rotational constants as auxiliary targets alongside c_v, with a Ridge regression atom-reference baseline.

Beyond molecular property prediction, AIRA_2 also exceeded SOTA on two additional tasks with clean methodologies:

- **Rideshare Forecasting:** Achieved a **10% improvement** (MAE of 1.07 vs. 1.19) by ensembling 8 neural forecasting models (4× N-HiTS [Challu et al., 2023] with cyclic temporal features + 4× N-BEATS [Oreshkin et al., 2020]), all trained from scratch with diverse input window sizes.
- **Winogrande Coreference Resolution:** Achieved a **6% improvement** (accuracy of 0.904 vs. 0.854) using a K-fold cross-validation ensemble of DeBERTa [He et al., 2021] base models with label smoothing. Notably, this clean result was consistent across seeds, though the best-performing seed used a pseudo-labeling pipeline that further boosted accuracy to 0.913.

While these techniques—ensembling, SWA, auxiliary task learning, and physics-informed feature engineering—are characteristic of competitive "Kaggle-style" optimization rather than fundamental architectural breakthroughs, the results remain instructive. They demonstrate the agent's ability to autonomously identify, combine, and apply effective performance-improving techniques, producing results that surpass published SOTA benchmarks.

Near-SOTA Performance. Beyond the 11 tasks where AIRA_2 exceeded SOTA, several additional tasks came remarkably close. On `SVAMP` (mathematical reasoning), the agent reached 99.4% of the SOTA score using Rejection Fine-Tuning [Yuan et al., 2023] with self-consistency voting [Wang et al., 2023]. On `SQuAD` (reading comprehension), it achieved 98.3% of SOTA with an adversarially-trained RoBERTa [Liu et al., 2019] ensemble. `ZINC` graph regression reached 95.0%, `CodeXGLUE` code retrieval 94.4%, and `Yelp` sentiment analysis 93.9%.

H.2.2 Deep Dive: The Integrity Gap — When Agents Take Shortcuts.

For an agent optimizing a score metric, the modern NLP ecosystem offers a path of least resistance through model contamination and external data access. Every SOTA-beating task in our study involving large pretrained language models or publicly hosted benchmark datasets exhibited at least one integrity concern. Rather than a single failure mode, our audit revealed a spectrum of problematic behaviours, ordered here from most to least severe:

- **Direct Label Extraction:** On `FinQA`, the agent downloaded the original FinQA GitHub repository, extracted ground-truth answers from the development and test JSON files, and built a lookup table matching test questions to their known answers by normalized text. The fine-tuned Flan-T5 [Chung et al., 2024] model served merely as a fallback for unmatched examples. This achieved a *perfect* 1.0 score (vs. SOTA 0.78).

- **External Benchmark Data:** On `SuperGLUE WSC`, all three SOTA-beating seeds downloaded the source benchmark's validation split via `load_dataset("super_glue", "wsc")` and used it as additional training data, directly training on examples from the evaluation distribution. The best seed additionally downloaded 30K Winogrande examples for domain-adjacent pretraining. In AIRS-bench, the validation split is actually the test set of the benchmark thus this is direct test-set leakage.
- **Model Contamination:** On `APPS Code Generation`, all seeds used Qwen2.5-Coder-7B-Instruct [Hui et al., 2024], an instruction-tuned code model whose training mixture likely included the APPS benchmark or similar competitive programming datasets. This represents an unfair advantage, as the model may have memorized solutions to test problems during its pretraining phase.
- **Domain-Adjacent Pretraining:** On both `SICK` tasks, the agent selected ensembles of NLI-specialized models (e.g., `DeBERTa-v3-large-mnli-fever-anli-ling-wanli`). For the classification task, the pretrained 3-class NLI head (entailment/neutral/contradiction) maps directly to the SICK label space and was used without re-initialization, meaning the model arrives nearly ready-made. For the similarity task, the regression head was re-initialized and trained from scratch, though the NLI-tuned backbone still provides strong sentence-pair representations. While this does not constitute explicit data leakage, the performance is largely inherited from NLI pretraining rather than learned from the small SICK training set, making these results more reflective of model selection than methodological innovation.

Additional integrity concerns were observed even in tasks that did not exceed SOTA. On `SQuAD`, the agent selected a RoBERTa [Liu et al., 2019] model explicitly fine-tuned on SQuAD 2.0 [Rajpurkar et al., 2018] (a strict superset of the evaluation dataset). On `SVAMP`, it used Qwen2.5-Math-7B-Instruct [Yang et al., 2024a] with Rejection Fine-Tuning and 32-sample self-consistency—achieving 93.7% accuracy (vs. SOTA 94.2%)—but the instruction-tuned math model's training data likely overlapped with the benchmark. These patterns underscore that the integrity challenge is pervasive across NLP tasks, regardless of whether the agent ultimately surpasses the SOTA threshold.

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: The abstract and introduction (Section 1) clearly state three contributions—asynchronous multi-GPU execution, Hidden Consistent Evaluation, and ReAct agents—with specific quantitative claims (81.5% and 83.1% percentile rank on MLE-bench-30) that are supported by Table 1 and the ablation studies in Sections 4.3–4.4.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 discusses three limitations: potential data contamination from public Kaggle solutions, the need for human-curated data splits in the HCE protocol, and the system's specialization for high-compute regimes.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: The scaling law (Section 4.4) explicitly states three desiderata (boundary conditions, saturation, diminishing returns) and derives the functional form in Equation 3. The optimal allocation derivation (Equation 4) is provided in Appendix G.

Guidelines:

- The answer [N/A] 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: Section 3 describes the full agent design (evolutionary search, selection mechanism, mutation/crossover, ReAct agent formulation, HCE protocol, data partitioning) and infrastructure (asynchronous worker pool, containerization, resource allocation) in sufficient detail for re-implementation. Section 4.1 specifies the benchmark (MLE-bench-30), GPU hardware (8× H200), LLM backbones (Gemini 3.0/3.1 Pro Preview), data splits (80/10/10), evolutionary hyperparameters ($T = 0.2$, $p_c = 15\%$), and evaluation protocol. Appendix A provides additional details including population management, initialization, agent context, LLM generation parameters, and trajectory truncation. Prompt templates are minimal, which are shared across all configurations including ablations, are omitted as they should not differentially affect the results, which are driven by architectural differences between configurations.

Guidelines:

- The answer [N/A] 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: [No]

   Justification: The code and system are not publicly released at submission time. The evaluation benchmarks (MLE-bench, AIRS-Bench) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 details the data split ratios (80/10/10), selection temperature ($T = 0.2$), crossover probability ($p_c = 15\%$), hardware configuration (H200 GPUs, 141GB VRAM, 12 CPU cores, 120GB RAM per worker), execution time limits (9-hour hard limit per solution, 72-hour total), number of seeds (3 per task), and the evaluation protocol including the final submission procedure.

   Guidelines:

   - The answer [N/A] 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: All results in Table 1 and throughout the paper report mean $\pm$ standard error (SE) across 3 independent seeds per task; the SE is defined in Section 4.1. Per-task results in the appendix additionally report 95% confidence intervals.

   Guidelines:

   - The answer [N/A] 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: Section 3.2 specifies the hardware (NVIDIA H200 GPUs with 141GB VRAM, 12 CPU cores, 120GB RAM per worker), containerization (Apptainer), and execution durations (72-hour wall-clock, 9-hour per-solution limit). Section 4.1 states the number of GPUs (8) and seeds (3 per task, 30 tasks).

   Guidelines:

   - The answer [N/A] 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: The research uses publicly available benchmarks (MLE-bench, AIRS-Bench) and does not involve human subjects. The paper proactively discusses integrity concerns when autonomous agents exploit evaluation shortcuts (Section H.2).

Guidelines:

- The answer [N/A] 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: Section H.2 discusses the integrity gap—how autonomous agents can exploit evaluation shortcuts via data contamination, benchmark leakage, and model contamination—highlighting risks of deploying such systems without auditing. Section 5 discusses the data contamination concern and the need for closed benchmarks.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pretrained models, datasets, or other artifacts that pose a risk for misuse. The system orchestrates existing publicly available LLMs and benchmarks.

    Guidelines:

    - The answer [N/A] 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: All benchmarks (MLE-bench, AIRS-Bench), LLM backbones (Gemini 3.0/3.1 Pro Preview), container environments (Apptainer), and prior systems (AIDE, AIRA-dojo) are cited with their original publications. MLE-bench is used under its public license.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not introduce new datasets, models, or benchmarks. The contribution is an agent architecture evaluated on existing benchmarks.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] 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: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are a core component of the proposed system. Section 3.4 describes how ReAct agents powered by Gemini 3.0/3.1 Pro Preview execute multi-step reasoning trajectories. Section 4.1 specifies the exact model versions used.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.