LEARNING THE ARTS OF SEARCH FOR AUTO ML

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated machine learning (ML) research can be formulated as an iterative search over hypotheses and their code implementations. Contemporary methods navigate this space with heuristics such as MCTS that rank nodes by score alone, conflating the merit of a hypothesis with the quality of its execution. As a result, a promising hypothesis with a preliminary implementation may rank below a modest one that is well tuned. We propose **A**gentic **R**easoning for **T**ree **S**earch (ARTS), where a reasoning language model inspects prior execution logs, diagnoses whether failures arise from faulty implementations or poor hypotheses, and selects which hypothesis to build on next. As the search progresses, the model suffers reasoning degradation due to longer contexts. In response, ARTS* learns from the previously explored hypotheses using test-time RL. Across 22 tasks from MLGym and MLEBench, ARTS outperforms leading methods by a mean normalized score of 12.0%. With test-time training, a Qwen3-4B scientist matches or exceeds an o3 scientist on 6 of 12 MLGym tasks. Notably, on MetaMaze, ARTS* learns to rediscover the human-best recurrent-memory solution that heuristic methods prune away.

1 INTRODUCTION

Research is an iterative *search process* where hypotheses are proposed, implemented in code, trained, evaluated, and revised. Automating this loop has a long history, from the Robot Scientist (King et al., 2009) to language-model agents for chemistry (Boiko et al., 2023), program search in combinatorics (Romera-Paredes et al., 2024), and recent systems for algorithm discovery and ML research (Novikov et al., 2025; Gottweis et al., 2025; Lu et al., 2024; Karpathy, 2026). Many of these systems achieve strong performance by scaling the number of candidate solutions and repeatedly executing experiments (Romera-Paredes et al., 2024; Novikov et al., 2025). In ML research, however, each experiment can be costly as a hypothesis must be translated into code, debugged, trained, and evaluated (Toledo et al., 2025; Nathani et al., 2025; Chan et al., 2024; Wijk et al., 2025).

Existing approaches broadly follow linear, tree-based, or evolutionary search. Linear methods (Nathani et al., 2025; Huang et al., 2024; Karpathy, 2026) repeatedly build on the latest hypothesis and can struggle to expand a prior hypothesis once the search enters an unpromising direction (Qin et al., 2025). Tree-based (Toledo et al., 2025; Jiang et al., 2025; Chi et al., 2024; Liang et al., 2026; Chen et al., 2026) and evolutionary (Novikov et al., 2025; Sharma, 2025; Lange et al., 2025; Weng et al., 2026) methods alleviate this limitation by revisiting earlier candidates, but the decision to revisit them is typically based on an observed score. This score conflates the quality of the underlying hypothesis with the outcome of its execution in code where a promising idea with an incomplete or poorly tuned implementation can be discarded in favor of a weaker idea that happens to be better optimized. For example, an untuned Transformer may initially underperform an LSTM, causing a score-based algorithm to continue exploiting the LSTM branch despite the potential of Transformers.

A natural alternative is to reason directly about which hypothesis is worth pursuing, which introduces two further challenges. First, reasoning models can exhibit diversity collapse (Zhang et al., 2025), while effective research requires balancing exploitation of promising ideas with exploration of novel ones (Herr et al., 2025; Yao et al., 2023a). Second, reasoning models struggle with increasing context length of the search history composed of hypotheses, code, execution traces, and experiment logs (Liu et al., 2024; Hsieh et al., 2024). An effective agent should reason over prior logs to decide the next experiment, but this ability decreases as the context increases.

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

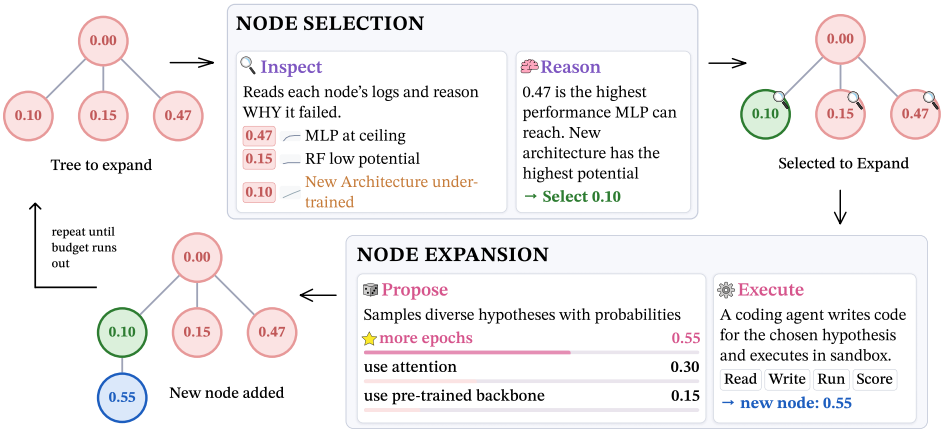


Figure 1: **A single search step of ARTS.** The `scientist` inspects candidate nodes, reasons about why each scored as it did, and selects the most promising rather than the highest-scoring one; here the 0.10 node, whose low score comes from under-training (Node selection). It proposes diverse hypotheses with verbalized probabilities, one is sampled, and the `executor` implements and scores it (Node expansion); the new node (0.55) joins the tree. Red: candidates; green: selected; blue: new.

We propose **Agentic Reasoning for Tree Search (ARTS)**, which reasons over the search history to guide exploration of new hypotheses (Fig. 1, Alg. 1). ARTS uses a large reasoning model to inspect any node's hypothesis, execution traces, and outcomes, and determine whether poor performance reflects a flawed hypothesis or a correctable implementation failure. By disentangling hypothesis quality from execution quality, ARTS can revisit promising ideas that score-based methods may prematurely discard. ARTS then selects which node to expand and proposes diverse hypotheses for the next experiment using Verbalized Sampling (Zhang et al., 2025). Across MLGym and MLEBench, ARTS improves the mean normalized score by 12.0% over leading algorithms.

To improve the scientist's reasoning over a long search history, we introduce ARTS*, which trains the scientist with RL at test time on its prior search trajectories (Sun et al., 2020; 2025) (Alg. 2). This makes the scientist learn from the already explored hypotheses and generate better ones. ARTS* trains LoRA (Hu et al., 2022) adapters with GRPO (Shao et al., 2024) where reward is based on the validation performance. With test-time training, a Qwen3-4B scientist matches or exceeds an o3 scientist on 6 of 12 MLGym tasks (Table 2). Remarkably, on MetaMaze, ARTS* rediscovers an LSTM-based solution and outperforms prior heuristic-search methods, which prune this branch after early instability, as well as the o3 scientist, which drifts away from it as the search history grows.

The contributions of this paper are:

- **Agentic Reasoning for Tree Search (ARTS)**, a search algorithm in which a Large Reasoning Model reasons over prior hypotheses, implementations, execution traces, and outcomes to select which node to expand, rather than relying on validation score alone.
- ARTS*, which **test-time trains** the scientist on the hypotheses it has explored, improving its ability to reason over and search the hypothesis space.
- An evaluation on MLGym and MLEBench showing that **ARTS outperforms leading methods** on 16 of 22 tasks and improves the mean normalized score by 12.0%. With ARTS*, a Qwen3-4B scientist matches or exceeds an o3 scientist on 6 of 12 MLGym tasks.

2 Related Work

Automated ML research is search over hypotheses, code, and execution traces; MLGym (Nathani et al., 2025), MLEBench (Chan et al., 2024), RE-Bench (Wijk et al., 2025), and DiscoveryBench (Majumder et al., 2025) make each node a real training run, and methods differ in how they choose the next node. Linear agents (ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), AI Scientist v1 (Lu et al., 2024), AutoResearch (Karpathy, 2026)) follow one trajectory and backtrack only within the current context (Guo et al., 2025; Qin et al., 2025; Liu et al., 2024; Hsieh et al., 2024). Tree search (AIDE (Jiang et al., 2025), AIRA (Toledo et al., 2025), AI Scientist v2 (Yamada et al., 2025))

selects nodes by greedy value or UCT, and evolutionary search (FunSearch (Romera-Paredes et al., 2024), AlphaEvolve (Novikov et al., 2025), OpenEvolve (Sharma, 2025), MLEvolve (Du et al., 2026)) by fitness; both conflate hypothesis quality with implementation quality, and mutation and crossover are brittle for tightly coupled ML pipelines. ARTS replaces heuristic based selection with an LRM deployed as scientist that inspects execution traces, reasons which hypothesis to build up on, maintains proposal diversity via verbalized sampling (Zhang et al., 2025), ARTS* trains the scientist on the explored hypothesis using test-time training (Sun et al., 2020; 2025; Akyürek et al., 2025; Yuksekgonul et al., 2026; Si et al., 2026) so that its reasoning does not degrade as the search history grows. Appendix K gives the extended discussion.

3 Research as a Search Problem

Research tasks. For a Machine Learning research task, we provide the agent with an initial workspace $\mathcal{W}_0$ (task instructions, data access, and an evaluation harness), an evaluation metric m, a baseline score s_0, and a wall-clock budget T; we write $\mathcal{P} = (\mathcal{W}_0, m, s_0, T)$. The code edits are tested inside a sandbox environment which queries the evaluator to obtain the execution trace and a validation score.

Experiments and search histories. For each research problem, the agent incrementally builds a search tree: at search step t (the wall-clock budget T bounds the run) it holds $\mathcal{G}_t = (V_t, E_t)$, where every node $v \in V_t$ is one evaluated experiment and stores a parent p_v, a hypothesis h_v, a code state c_v. The code state c_v is built from the parent's code state c_{p_v} and uses information like logs ℓ_v, score s_v, depth d_v, and count of edges from root v. At step t the agent takes action $a_t = (p_t, h_t, c_t)$ to expand the parent $p_t \in V_t$ with the hypothesis h_t, and its implementation c_t starting from c_{p_t}. The evaluator runs c_t and returns the logs ℓ_t and the score s_t, and the new node $v_t = (p_t, h_t, c_t, \ell_t, s_t)$ is added to give $\mathcal{G}_{t+1}$. We call the component that proposes h_t as the `scientist` and the one that writes c_t as the `executor`. Prior methods (Toledo et al., 2025; Du et al., 2026; Karpathy, 2026) use one agent for both proposing the hypothesis h_t and writing the code c_t.

Hypothesis quality and execution quality. The score of a node depends on both its hypothesis and its implementation. We write $s(h, c)$ for the score of implementation c of hypothesis h, and define the quality of a hypothesis as the best score any implementation of it reaches,

$$s^*(h) = \max_c s(h, c). \tag{1}$$

The execution gap of node v is $s^*(h_v) - s(h_v, c_v)$. It is large for a preliminary execution (an untuned model, a short run, a buggy pipeline). Score-based selection observes only $s(h_v, c_v)$ and ranks a strong hypothesis with a weak implementation below a weaker one with a good implementation. Since $s^*(h)$ is not observable, ARTS asks the scientist to judge using the code and execution logs to check whether a low score comes from h_v or from c_v (§4.1).

4 Agentic Reasoning for Tree Search

We present **ARTS** (**A**gentic **R**easoning for **T**ree **S**earch) with two design choices. First, the scientist and the executor are different models, since choosing a research direction and writing correct code need different abilities. Second, the scientist chooses both the parent p_t and generates the hypothesis h_t after looking at the search tree and the persistent memory block. It can inspect any node's code and logs. It then uses a modified version of verbalized sampling (Zhang et al., 2025) to keep its proposals diverse by listing K candidate hypotheses with verbalized probabilities. The system then draws one uniformly at random, which the executor implements for evaluation.

4.1 The Scientist

The `scientist` selects which node to expand and what hypothesis to try, both of which need reasoning over several past experiments. For example the model would conclude that the bottleneck is the loss function rather than model capacity after observing that more capacity did not improve the validation accuracy. The default choice, therefore, is to use a large reasoning model as the `scientist` for its multi-step reasoning (OpenAI, 2025; Chollet et al., 2025; Glazer et al., 2025; Rein et al., 2023). Its components follow Algorithm 1.

Algorithm 1: ARTS search.

Input: Task $\mathcal{P}$, time budget T, `scientist`, `executor`
Initialize: $t \leftarrow 0$; tree $\mathcal{G}_0$ = (code c_{base}, score s_{base})
AUDIT data, code, and metric // no expansion
while time $< T$ **do**
  // inspect the tree to fetch logs
  $\ell_t \leftarrow \text{INSPECT}(\mathcal{G}_t, \mathcal{P})$
  // select parent, propose K hypotheses
  $p_t, \{(h_t^k, \pi_t^k)\}_{k=1}^K \leftarrow \text{SCIENTIST}(\ell_t, \text{mem.})$
  // external sampler draws one hypothesis
  $h_t \leftarrow \text{SAMPLE}(\{(h_t^k, \pi_t^k)\}_{k=1}^K)$
  // implement and validate
  $c_t, s_t \leftarrow \text{EXECUTE}(p_t, h_t, c_{p_t})$
  // write one insight to memory
  $\text{mem.} \leftarrow \text{UPDATEMEMORY}(\text{mem.}, h_t, c_t, s_t)$
  // add the new node to the tree
  $\mathcal{G}_{t+1} \leftarrow \mathcal{G}_t \cup \{v_t = (p_t, h_t, c_t, s_t)\};\ t \leftarrow t+1$
end while
return best validated node

Algorithm 2: ARTS* with test time training.

Input: Task $\mathcal{P}$, group size B, `scientist` π_θ, `executor`
Initialize: $\mathcal{G}$ = (c_{base}, s_{base}) LoRA adapters.
for each GRPO step t **do**
  $\ell_t \leftarrow \text{INSPECT}(\mathcal{G}_t, \mathcal{P})$
  // parent selection and hypothesis sampling
  $\{(p_i, \{(h_i^k, \pi_i^k)\}_{k=1}^K)\}_{i=1}^B \sim \pi_\theta(\mathcal{G}_t, \ell_t, \text{mem.})$
  $h_i \leftarrow \text{SAMPLE}(\{(h_i^k, \pi_i^k)\}_{k=1}^K)$ for each i
  $c_i, s_i \leftarrow \text{EXECUTE}(p_i, h_i, c_{p_i})$ for each i
  // percentile reward
  $r_i \leftarrow R(s_i, \mathcal{G}_t)$
  // GRPO update of the LoRA adapters
  $\theta \leftarrow \text{GRPO}(\theta, \{(p_i, h_i, r_i)\}_{i=1}^B)$
  // add rollouts to the tree
  $\mathcal{G}_{t+1} \leftarrow \mathcal{G}_t \cup \{(p_i, h_i, c_i, s_i)\}_{i=1}^B$
end for
return trained scientist $\pi_{\theta'}$, best validated node

Node Inspection. A node's score alone does not provide enough information as to why a node succeeded or failed. The INSPECT tool returns the code, training curves, and any logs from the prior nodes. The `scientist` is free to inspect as many nodes as it wants. This helps the `scientist` to reason about why a node failed and whether to expand on it or not. For instance, inspecting that the current sparse reward definition leads to no positive rewards would help the model explore other reward shaping techniques over just scaling model size.

Baseline Audit. Many tasks have structure that can be exploited to get to a good solution (e.g. Vesuvius has few labeled data but plentiful unlabeled data, which might enable self supervised learning). We reserve the first R scientist calls as AUDITS. In this stage no node is expanded and no code changes. The `scientist` reads the data, baseline code, and evaluation script and must state the structure of the metric and the largest information-loss step in the baseline. We use $R = 3$; on hard tasks the audit finds the main bottleneck, for example that the 65 CT z-slices of Vesuvius carry information non-uniformly (sweep in Appendix Table 12).

Hypothesis sampling. Exploring diverse hypothesis is essential in research. If we simply ask the scientist for the next hypothesis, it returns the most probable continuation, usually a small learning-rate tweak. Hence, we use a variant of verbalized sampling (Zhang et al., 2025) where it enumerates K candidates h_t^k with probabilities π_t^k and an external sampler draws one uniformly at random. This is done due to two reasons, first verbalized sampling's probabilities are not calibrated and are only to elicit diverse candidates. Second, if we ask the scientist itself to sample one, it tends to pick the first. At deeper nodes we ask for candidates from the tail. (For more information, look at Appendix A.6).

Memory. For each experiment, the scientist adds a short insight (score, failure mode) to an editable memory. This memory is retrieved to select, inspect or expand a node or to propose a hypothesis. This is complimentary to inspect which provides full logs of a single node. The scientist is prompted to avoid adding duplicate information and we do not specify a fixed memory size(see App. B).

4.2 THE EXECUTOR

The `executor` is a coding agent that takes h_t and the parent's validated code c_{p_t} and writes c_t. For this role, it is preferred to use an agentic language model (DeepMind, 2025), which is fast, inexpensive, and strong at code generation. It has three tools, READ_FILE, WRITE_FILE, and VALIDATE (to run the code in the sandbox and return the validation score). The `executor` edits and validates until a valid score is returned or the action budget is exhausted. It is prompted to implement h_t exactly and without this rule it substitutes hypotheses of its own. The node joins the tree with its score s_t (null if no valid submission).

4.3 ARTS*: TEST-TIME TRAINING THE SCIENTIST

The search tree $\mathcal{G}_t$, especially on hard problems like drug discovery, can grow exponentially with depth: a tree with depth 8 and branching factor 3 has over 6000 nodes. Reasoning degrades over such long contexts (Liu et al., 2024; Hsieh et al., 2024). To solve this, ARTS* learns from the

Table 1: Per-task results on 22 MLGym and MLEBench tasks: mean ± standard error of the best score over 3 runs; arrows give the direction of each metric. Prior works run one model as both scientist and executor, either Gemini 3 Flash (as in their original implementations) or o3; ARTS is shown with three scientist/executor pairs, o3/Gemini 3 Flash being the configuration used elsewhere in the paper. Metrics are listed in Appendix C. Human Best is Kaggle top-1 or the best published result, a reference scale. Bold/highlight: best method per row, ties included (UCB ablation: Appendix Table 5).

		Prior works, Gemini 3 Flash			*Prior works, o3*			*ARTS (scientist / executor)*			
Task	**Base**	**Lin.**	**AIRA**	**MLE.**	**Lin.**	**AIRA**	**MLE.**	**Gem/Gem**	**o3/o3**	**o3/Gem**	**Human**
MLGym											
Titanic ↑	0.766	$0.951_{\pm.004}$	$0.944_{\pm.001}$	$0.946_{\pm.002}$	$0.955_{\pm.004}$	$0.949_{\pm.002}$	$0.951_{\pm.003}$	$0.958_{\pm.005}$	$0.979_{\pm.005}$	**0.984**$_{\pm.004}$	0.830
CIFAR-10 ↑	0.497	$0.956_{\pm.002}$	$0.964_{\pm.002}$	$0.959_{\pm.003}$	$0.961_{\pm.003}$	$0.966_{\pm.002}$	$0.963_{\pm.003}$	$0.962_{\pm.003}$	$0.965_{\pm.004}$	**0.971**$_{\pm.017}$	0.994
F-MNIST ↑	0.848	$0.946_{\pm.006}$	$0.947_{\pm.000}$	$0.948_{\pm.001}$	$0.949_{\pm.004}$	$0.950_{\pm.001}$	$0.951_{\pm.002}$	$0.950_{\pm.002}$	$0.953_{\pm.002}$	**0.958**$_{\pm.001}$	0.968
House Price ↑	0.880	$0.940_{\pm.000}$	**0.944**$_{\pm.001}$	$0.943_{\pm.001}$	$0.935_{\pm.002}$	$0.938_{\pm.001}$	$0.937_{\pm.002}$	$0.935_{\pm.002}$	$0.939_{\pm.003}$	$0.939_{\pm.003}$	0.990
MNLI ↑	52.51	$84.42_{\pm0.05}$	$83.77_{\pm0.05}$	$84.26_{\pm0.08}$	$84.51_{\pm.09}$	$84.03_{\pm.07}$	$84.38_{\pm.10}$	$84.12_{\pm.11}$	$84.40_{\pm.3}$	**84.71**$_{\pm0.49}$	92.50
Lang. Model. ↓	4.673	$3.986_{\pm.169}$	$4.673_{\pm.000}$	$4.015_{\pm.146}$	$3.950_{\pm.12}$	$4.850_{\pm.21}$	$4.670_{\pm.00}$	$4.621_{\pm.21}$	$3.91_{\pm.10}$	**3.827**$_{\pm.088}$	3.500
Battle of Sexes ↑	1.023	$1.448_{\pm.001}$	$1.442_{\pm.001}$	$1.446_{\pm.000}$	$1.450_{\pm.002}$	$1.420_{\pm.003}$	$1.410_{\pm.002}$	$1.620_{\pm.09}$	$1.98_{\pm.02}$	**2.000**$_{\pm.000}$	1.667
Prisoner's D. ↑	2.372	$2.453_{\pm.102}$	$2.501_{\pm.129}$	$2.635_{\pm.012}$	$2.512_{\pm.09}$	$2.588_{\pm.11}$	$2.671_{\pm.02}$	$2.640_{\pm.02}$	$2.80_{\pm.15}$	**2.857**$_{\pm.225}$	3.000
Blotto ↑	−0.248	$-0.076_{\pm.327}$	$0.247_{\pm.003}$	$0.250_{\pm.001}$	$0.250_{\pm.001}$	$0.250_{\pm.001}$	$0.250_{\pm.001}$	$0.254_{\pm.002}$	**0.255**$_{\pm.002}$	$0.249_{\pm.001}$	0.500
MountainCar ↑	33.79	$45.44_{\pm6.27}$	$80.82_{\pm11.75}$	$84.84_{\pm6.38}$	$46.02_{\pm5.1}$	$75.61_{\pm8.7}$	$76.68_{\pm6.0}$	$93.19_{\pm1.8}$	$94.2_{\pm1.5}$	**95.73**$_{\pm.69}$	99.00
Breakout ↑	48.82	$64.03_{\pm10.75}$	$57.94_{\pm3.37}$	**83.28**$_{\pm3.84}$	$65.10_{\pm7.4}$	$74.61_{\pm4.2}$	$71.58_{\pm3.9}$	$77.58_{\pm2.7}$	$77.9_{\pm2.6}$	$78.00_{\pm2.23}$	100.00
Meta Maze ↑	15.73	$46.80_{\pm2.61}$	$36.42_{\pm6.47}$	$45.35_{\pm3.27}$	$26.48_{\pm3.3}$	$39.11_{\pm5.0}$	$30.00_{\pm4.1}$	$49.97_{\pm1.6}$	$50.4_{\pm1.4}$	**51.20**$_{\pm1.03}$	52.50
MLEBench											
Spaceship ↑	0.000	$0.825_{\pm.006}$	0.831	$0.834_{\pm.001}$	$0.829_{\pm.004}$	$0.833_{\pm.002}$	$0.835_{\pm.001}$	$0.832_{\pm.002}$	$0.834_{\pm.002}$	**0.836**$_{\pm.001}$	0.828
Nomad 2018 ↓	1.000	$0.063_{\pm.001}$	**0.062**$_{\pm.001}$	$0.063_{\pm.000}$	$0.065_{\pm.001}$	$0.065_{\pm.001}$	$0.066_{\pm.001}$	$0.065_{\pm.001}$	$0.064_{\pm.001}$	$0.064_{\pm.001}$	0.051
Jigsaw Toxic ↑	0.500	**0.980**$_{\pm.000}$	**0.980**$_{\pm.000}$	**0.980**$_{\pm.000}$	$0.979_{\pm.000}$	$0.979_{\pm.000}$	$0.979_{\pm.000}$	$0.979_{\pm.000}$	**0.980**$_{\pm.000}$	**0.980**$_{\pm.000}$	0.989
APTOS 2019 ↑	0.000	$0.926_{\pm.001}$	$0.922_{\pm.006}$	$0.914_{\pm.004}$	$0.927_{\pm.002}$	$0.924_{\pm.005}$	$0.919_{\pm.004}$	$0.918_{\pm.005}$	$0.921_{\pm.006}$	**0.930**$_{\pm.004}$	0.936
Plant Path. ↑	0.500	$0.994_{\pm.001}$	$0.997_{\pm.000}$	**0.998**$_{\pm.000}$	$0.992_{\pm.001}$	$0.993_{\pm.001}$	$0.994_{\pm.001}$	$0.993_{\pm.002}$	$0.994_{\pm.002}$	$0.995_{\pm.003}$	0.984
Histopath. ↑	0.500	$0.990_{\pm.001}$	**0.995**$_{\pm.000}$	$0.994_{\pm.000}$	$0.991_{\pm.001}$	$0.993_{\pm.001}$	$0.994_{\pm.000}$	$0.992_{\pm.001}$	$0.994_{\pm.001}$	**0.995**$_{\pm.000}$	1.000
Vesuvius ↑	0.000	$0.479_{\pm.067}$	$0.309_{\pm.112}$	$0.549_{\pm.011}$	$0.122_{\pm.05}$	$0.327_{\pm.09}$	$0.443_{\pm.02}$	$0.114_{\pm.04}$	$0.486_{\pm.04}$	**0.551**$_{\pm.021}$	0.831
Kuzushiji ↑	0.000	$0.894_{\pm.040}$	$0.872_{\pm.026}$	**0.921**$_{\pm.009}$	$0.820_{\pm.03}$	$0.815_{\pm.02}$	$0.831_{\pm.01}$	$0.812_{\pm.03}$	$0.840_{\pm.03}$	$0.843_{\pm.034}$	0.950
HMS Brain ↓	1.462	$0.543_{\pm.055}$	$0.550_{\pm.020}$	$0.583_{\pm.013}$	$1.390_{\pm.06}$	$1.140_{\pm.04}$	$0.910_{\pm.03}$	$0.461_{\pm.010}$	**0.455**$_{\pm.012}$	$0.499_{\pm.008}$	0.272
RSNA ↑	0.500	$0.638_{\pm.005}$	$0.649_{\pm.014}$	$0.656_{\pm.011}$	$0.644_{\pm.006}$	$0.653_{\pm.012}$	$0.661_{\pm.010}$	$0.649_{\pm.010}$	$0.655_{\pm.02}$	**0.673**$_{\pm.021}$	0.621

previously explored hypotheses by test-time training (TTT) the `scientist` on them, updating LoRA adapters (Hu et al., 2022) at regular intervals (Alg. 2). Adapters are trained from scratch on each task using that task's tree alone. Transfer across tasks is outside the scope of this paper.

Training. Prior work (Yuksekgonul et al., 2026) explored TTT for hypothesis generation, but the action is only the hypothesis and node selection is heuristic. The agent makes two decisions per step– (i) which node p_t in the search tree to expand from, and (ii) what hypothesis h_t to test. Given the state $\mathcal{G}_t$ (§3), the `scientist` generates the action $a_t = (p_t, h_t)$. We sample B such actions from $\mathcal{G}_t$ and compute a reward for each. The B rewards form a GRPO (Shao et al., 2024) group used to update the adapters. Since each action creates a new node, nodes are appended to the tree before the next batch is sampled. This teaches the model which parent to pick and what hypothesis to explore for the largest gains in validation score. We define the reward based on the score s_t that the action $a_t = (p_t, h_t)$ receives upon execution. We use the single-step percentile-based rewards:

$$r_t = \begin{cases} -0.5 & s_t = \text{null}, \\ -0.2 & s_t < s_{\text{base}}, \\ 0 & s_{\text{base}} \le s_t < s_{70}(\mathcal{G}_t), \\ 1 & s_t \ge s_{70}(\mathcal{G}_t), \end{cases} \tag{2}$$

where $s_{70}(\mathcal{G}_t)$ is the 70th percentile of the last 64 valid scores up to step t. This is denser than rewarding only the best node, since the top 30% of nodes get a positive reward rather than just the maximum. The adaptive threshold rises as the policy improves, preventing reward saturation.

5 Experiments

Tasks. We evaluate on 22 tasks from MLGym (Nathani et al., 2025) (regression, language modeling, vision, RL, and game theory) and MLEBench (Chan et al., 2024) (Kaggle competitions; 4 hard, 3 medium, 3 easy, selected at random within each difficulty tier).[1] Each task provides the workspace $\mathcal{W}_0$ (§3) and a held-out test set. Human-best scores are Kaggle top-1 (MLEBench) or the best published result (MLGym; Appendix C).

[1]Not the entire suite of tasks because of compute and API constraints.

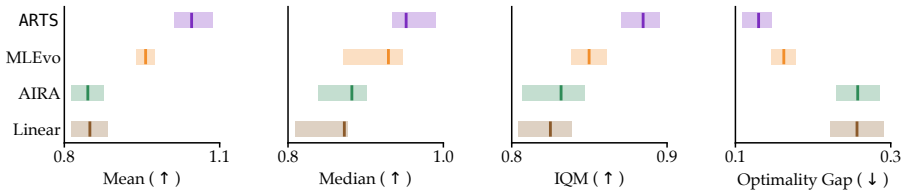


Figure 2: **Aggregate normalized performance**. Reliable metrics of Agarwal et al. (2021) with 95% stratified bootstrap intervals. Baselines use Gemini 3 Flash for both roles (o3 in Table 1).

Baselines. We compare ARTS with three prior search methods from § 2. Linear Search uses the AutoResearch prompt (Karpathy, 2026), Tree Search uses AIRA (MCTS) (Toledo et al., 2025), and Evolutionary Search uses MLEvolve Du et al. (2026), and methods receive same executor, container, action set, and runtime budgets. Each baseline uses the same model for both scientist and executor.

Models. We use OpenAI o3 (OpenAI, 2025) and Gemini 3 Flash (DeepMind, 2025) as both the scientist and the executor. We also experiment with using different models for scientist and executor based on their capabilities. Since o3 is better at reasoning we use that as the scientist and since Gemini 3 Flash is faster, we use that for executor. The prompts for both the scientist and executor are in Appendix D. ARTS* uses Qwen3-4B-Instruct (Yang et al., 2025) as the scientist.

Evaluation. All methods run inside Apptainer containers with the same action set, executor sandbox, and 8-hour wall-clock budget (time rather than node count, as in prior work (Karpathy, 2026; Toledo et al., 2025; Du et al., 2026; Chan et al., 2024)). Since agents can modify evaluation scripts, we restore them before validation (Appendix B). We report the mean and standard error over 3 independent runs of the best validation score reached in a run. Scores come from the task's restored evaluation script on the held-out validation split provided with the task; the search optimizes the same score. Human-best values are leaderboard or published test-set results and serve as a reference scale, not an upper bound (every method exceeds it on Titanic, Plant Pathology, and RSNA). Each method is given one 40 GB A100 GPU for 8 hours to train any downstream model during the search process. ARTS* uses two more GPUs (Qwen3-4B server and adapter trainer), 24 GPU-hours in total (costs in Appendix Table 3).

6 Results

Does ARTS search better than prior methods? It is best on 16 of 22 tasks under the same budget, and on 17 of 22 with the baselines on o3 (§6.1). These gains come from failure attribution, the diversity and quality of its hypotheses (§6.2). Finally, test-time training helps. A Qwen3-4B scientist matches or exceeds o3 on 6 of 12 MLGym tasks (§6.3).

6.1 How does ARTS compare to prior methods?

ARTS outperforms existing methods. Table 1 reports per-task scores. ARTS with an o3 scientist and a Gemini 3 Flash executor is the best method on 16 of 22 tasks and improves the mean normalized score by 12.0% over the strongest baseline. The gains are highest on the harder MLEBench and MLGym tasks, where each hypothesis is an expensive training run and an early poor choice wastes the time budget; on RSNA Brain Tumor, ARTS exceeds the human-best reference. House Price and Jigsaw Toxic are near saturation for every method.

To control for the choice of model, we compare each method with the same scientist and executor (Table 1). The baselines do not use separate agents for both roles, so they cannot be run with an o3 scientist and a Gemini 3 Flash executor; we run each of them with o3 and with Gemini 3 Flash. With o3 for both roles, ARTS is best on 17 of 22 tasks and improves the mean normalized score by 19.4% over the strongest o3 baseline; with Gemini 3 Flash for both roles, it is best on 8 of 22 and trails the strongest Gemini 3 Flash baseline by 2.8%. The gains with Gemini 3 Flash are small because reasoning-based selection needs a strong reasoner such as o3; as frontier models improve at reasoning, we expect further gains from our approach. ARTS with its o3 scientist and Gemini 3 Flash executor still wins 21 of 22 tasks against the baselines run on o3, losing only Blotto by 0.001. Replacing reasoning with UCB in ARTS, all else equal, lowers the score on all 22 tasks (MountainCar: 95.73 to 65.7; Appendix Table 5), indicating that the gains come from reasoning-based selection.

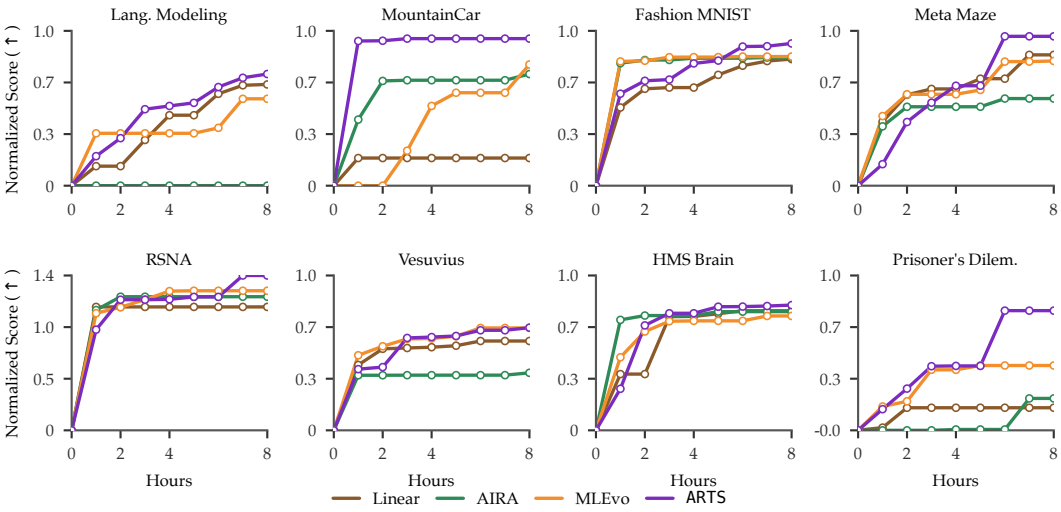


Figure 3: Best normalized score against wall-clock time under the same 8-hour budget (mean of 3 runs); all methods share the Gemini 3 Flash executor. Normalized score is $(s_{\text{method}} - s_{\text{base}})/(s_{\text{human}} - s_{\text{base}})$: 0 matches the starter code and 1 the human-best reference, for lower-is-better metrics too, since numerator and denominator change sign together.

Figure 2 aggregates per-task scores with the reliable metrics of Agarwal et al. (2021). ARTS has the highest inter-quartile mean, which trims the top and bottom 25% of runs and so cannot be inflated by a few lucky outliers. It also has the smallest optimality gap to the human-best reference, which shows the improvement is reliable.

Figure 3 plots the best normalized score against wall-clock time under the same 8-hour budget. We observe that ARTS typically trails for the first one or two hours, catches up mid-run, and finishes above the strongest contemporary method. This pattern of a slow start followed by a higher ceiling shows that ARTS spends its early budget exploring broadly instead of committing onto the first locally good hypothesis, as opposed to other methods,. leading to better utilizaton of the inference budget.

6.2 Why does ARTS outperform prior methods?

We trace ARTS's gains to three factors: *failure attribution*, the *diversity* of hypotheses it explores, and the *quality* of each hypothesis, as described below.

Finding 1. *ARTS does not* ***prematurely abandon a promising hypothesis.***

Prior tree-search methods select on the score alone, so a coding bug, bad hyperparameters, under-training, and a poor hypothesis all look the same and the search pivots away from sound ideas. ARTS has the `scientist` inspect a node's code and logs and classify a low score as implementation-wrong or idea-wrong; the next experiment stays on the same axis with corrected instructions in the first case and pivots in the second.

For instance, on the HMS Brain Activity task (Appendix Fig. 8; lower KL is better; baseline = 1.462), ARTS drafts a pretrained ResNet-50 over 3-channel log-mel spectrograms at depth 1 and reaches **KL 0.557**. The next two children, a backbone swap to EfficientNet-B0 (0.702) and an added SpecAugment (0.565), both regress. A score-only selection algorithm, AIRA for instance, reads these regressions as evidence that the spectrogram-CNN family is exhausted, and its next two drafts pivot to hand-crafted features, and the childrenhi of those pivots diverge catastrophically to KL 4.08 and 9.43 while the original spectrogram-CNN family is never refined. *ARTS instead re-reads the two regressed nodes' training logs, observes that the validation KL is still decreasing at the last epoch in both, and classifies them as implementation-wrong* (under-trained under heavy augmentation), retaining the ResNet-50 + spectrogram family. The next few expansions improve the KL further to 0.516; and continued refinement on the same family, that AIRA abandoned early, ultimately reaches

KL 0.467 in its best run, making ARTS the best method on this task. APTOS and Vesuvius show the same pattern, with trees and node-level diagnoses in Appendix E.

Finding 2. *ARTS distributes its expansions across **diverse hypothesis** families compared to score-based methods that concentrate around a small set of hypotheses.*

Exploring diverse hypotheses is important in open-ended search problems like ML research: refining the approach that scored best first is not guaranteed to be globally optimal. Score-only methods concentrate on a small set of leading hypotheses and explore only their close variants. In contrast, ARTS proposes a wider range of diverse hypotheses, widening the explored set substantially more than score-only methods.

Labeling every proposed hypothesis into one of eight axes with a coding agent (GPT-5.5), ARTS covers **4.43** axes per run against 4.05 for MLEvolve, only 1.05 above its number of initial draft nodes, and its proposal entropy over the axes is **1.73** against 1.35 for MLEvolve and 1.48 for AIRA. For human validation, we hand-label 75 expansions across 15 runs, five per method, as either a completely new hypothesis, for instance architectural changes, or a minor change within the current hypothesis, like hyperparameter tuning. **45%** of ARTS's expansions open a completely new hypothesis, against 3% for AIRA and 0% for MLEvolve. On these 75 expansions, the GPT-5.5 axis labels agree with the human labels with Cohen's $\kappa = 0.83$. Finally, the within-run pairwise TF-IDF distance between hypotheses averages **0.72** for ARTS, 0.23 for AIRA, and 0.14 for MLEvolve, a lexical rather than semantic measure.

The Vesuvius ink-detection search trees (Appendix Fig. 9) illustrate this qualitatively. Compared with AIRA, ARTS holds five distinct axes as siblings at the root (architecture, slice representation, loss, training data, augmentation), identifies that adding the second fragment is a promising strategy ($F_{0.5} = 0.479$), and deepens that one branch with augmentation to reach $\mathbf{0.575}$ in its best run. Whereas, AIRA focuses on tweaking architectures with a single deep chain.

Finding 3. *ARTS spends the exploration budget on **higher quality hypotheses** compared to score-only methods.*

Exploring many directions helps only when the hypotheses explored are of high quality. In ARTS, before proposing the next hypothesis, the `scientist` reads previous nodes' code and training logs to identify a specific failure and proposes a hypothesis that addresses it. Each candidate is therefore well reasoned and high quality, rather than a random mutation of the current best code or an extension of one of the highest-scoring branches.

The MetaMaze trees (Appendix Fig. 10; baseline reward 15.73, partial-observability) demonstrate this. The problem requires both preserving 2D maze geometry at the input and recurrent memory. ARTS looks at the baseline logs and diagnoses that the baseline flattens the input, vectorising each 2D maze slice before the network sees it, which destroys the wall and goal geometry. *ARTS fixes the geometry first by using a CNN encoder (*23.03*), and richer egocentric observation (*48.57*); it then proposes an LSTM recurrent policy but abandons it after early instability and reaches* 51.20 *on MLP/PPO, whereas the test-time trained scientist follows the recurrent direction to* 53.0*, above the human-best reference.* AIRA applies the textbook PO-MDP prior, LSTM, as its first draft (48.04) without inspecting the input pipeline, then stacks larger nets on the LSTM with improper execution that destabilises the training (17.5–22.3) eventually leading the method to discard the LSTM family.

6.3 Does test-time training help ARTS* search efficiently?

Test-time training raises the mean normalized score of the Qwen3-4B scientist on MLGym from 0.72 to $\mathbf{1.01}$ (Table 2). It matches or exceeds the o3 scientist on 6 of 12 tasks. The largest gains are on Language Modeling (4.34 to $\mathbf{3.52}$, matching the human-best 3.50) and MetaMaze (53.0 against 52.5). On MetaMaze, prior methods propose LSTM/GRU policies, find them unstable, and discard them; ARTS with o3 identifies the need for memory and tries an LSTM but abandons it for high-scoring MLP/PPO. *The test-time trained scientist returns to the recurrent direction and derives a well-optimized LSTM solution* (Appendix Figure 11). The untrained o3 scientist loses this direction as its context grows, whereas the trained Qwen3-4B scientist does not. On MountainCar, ARTS reaches its peak reward in 4 expansions where AIRA needs a longer chain (Appendix I). Training on the trajectories also beats compacting them, the strategy AIRA (Toledo et al., 2025) uses to shorten a

Table 2: ARTS*: test-time training a Qwen3-4B scientist on its own search trajectories (MLGym). +Com: the same 4B scientist with the search history compacted instead of trained on. Mean ± standard error over 3 runs; best highlighted; MLEBench results in Table 1.

	Prior Works			*Ours*				
Task	Linear	AIRA	MLEvolve	ARTS (o3)	ARTS (4B)	ARTS (4B)+Com	ARTS* (4B)	Human Best
Titanic (↑)	$0.951_{\pm.004}$	$0.944_{\pm.001}$	$0.946_{\pm.002}$	$0.984_{\pm.004}$	$0.949_{\pm.009}$	$0.952_{\pm.006}$	**0.998**$_{\pm.002}$	0.830
CIFAR-10 (↑)	$0.956_{\pm.002}$	$0.964_{\pm.002}$	$0.959_{\pm.003}$	$0.971_{\pm.017}$	$0.957_{\pm.002}$	$0.960_{\pm.003}$	**0.982**$_{\pm.006}$	0.994
Fashion MNIST (↑)	$0.946_{\pm.006}$	$0.947_{\pm.000}$	$0.948_{\pm.001}$	**0.958**$_{\pm.001}$	$0.946_{\pm.002}$	$0.946_{\pm.002}$	$0.948_{\pm.007}$	0.968
House Price (↑)	$0.940_{\pm.000}$	**0.944**$_{\pm.001}$	$0.943_{\pm.001}$	$0.939_{\pm.003}$	$0.929_{\pm.001}$	$0.931_{\pm.002}$	$0.940_{\pm.001}$	0.990
MNLI (↑)	$84.42_{\pm0.05}$	$83.77_{\pm0.05}$	$84.26_{\pm0.08}$	**84.71**$_{\pm0.49}$	$83.33_{\pm.08}$	$83.41_{\pm.12}$	$84.01_{\pm1.81}$	92.50
Lang. Modeling (↓)	$3.986_{\pm.169}$	$4.673_{\pm.000}$	$4.015_{\pm.146}$	$3.827_{\pm.088}$	$4.34_{\pm.48}$	$4.21_{\pm.31}$	**3.518**$_{\pm.276}$	3.500
Battle of Sexes (↑)	$1.448_{\pm.001}$	$1.442_{\pm.001}$	$1.446_{\pm.000}$	**2.000**$_{\pm.000}$	$1.441_{\pm.000}$	$1.441_{\pm.000}$	$1.442_{\pm.000}$	1.667
Prisoner's Dilemma (↑)	$2.453_{\pm.102}$	$2.501_{\pm.129}$	$2.635_{\pm.012}$	**2.857**$_{\pm.225}$	$2.641_{\pm.001}$	$2.640_{\pm.003}$	$2.633_{\pm.318}$	3.000
Blotto (↑)	$-0.076_{\pm.327}$	$0.247_{\pm.003}$	$0.250_{\pm.001}$	$0.249_{\pm.001}$	$0.251_{\pm.002}$	$0.250_{\pm.001}$	**0.344**$_{\pm.33}$	0.500
MountainCar (↑)	$45.44_{\pm6.27}$	$80.82_{\pm11.75}$	$84.84_{\pm6.38}$	**95.73**$_{\pm.69}$	$56.65_{\pm1.69}$	$61.2_{\pm3.4}$	$91.94_{\pm2.83}$	99.00
Breakout (↑)	$64.03_{\pm10.75}$	$57.94_{\pm3.37}$	**83.28**$_{\pm3.84}$	$78.00_{\pm2.23}$	$48.82_{\pm.00}$	$52.3_{\pm2.9}$	$75.95_{\pm4.83}$	100.00
Meta Maze (↑)	$46.80_{\pm2.61}$	$36.42_{\pm6.47}$	$45.35_{\pm3.27}$	$51.20_{\pm1.03}$	$30.35_{\pm20.67}$	$33.9_{\pm9.8}$	**53.00**$_{\pm.57}$	52.50

long search history by summarizing older nodes: with the same Qwen3-4B scientist, compaction raises the mean normalized score only from 0.72 to 0.76, against 1.01 with test-time training (Table 2, +Com).

6.4 Ablations

Models and components. The executor matters even with the scientist fixed: in single-seed swap runs, normalized score rises from 0.20 to 0.72 on Language Modeling and from −0.16 to 0.30 on Vesuvius (Appendix Fig. 4). With the executor fixed, o3 is the only scientist that reliably moves beyond baseline on these tasks. With reasoning tokens and GPU time, one ARTS run costs $12.31 against $10.29–$13.68 for the baselines (Appendix Table 3). Removing the audit or verbalized sampling gives the largest drops, 0.809 to 0.470/0.506 on HMS and 0.887 to 0.573/0.614 on Kuzushiji (Appendix Fig. 5).

Test-time training. The percentile reward beats every alternative on all three reward-ablation tasks (mean normalized **1.03** against 0.29) and the continuous entropic objective of TTT-Discover on Language Modeling (Appendix A.7), with a rolling window of 64 scores (sweep in Appendix Table 13). Single-step GRPO beats tree-per-episode rollouts (**1.48** against 0.89), since one-step rewards Jain et al. (2025) give clearer credit assignment than multi-step trajectory rewards. Training does not collapse diversity: runs cover 6–10 strategy categories, and entropy rises from early to late rollouts on 3 of 4 tasks (Appendix Figs. 6 and 7).

Contamination. The MLEBench competitions predate the models' training cutoffs, but the same executor writes all code and the o3 baselines in Table 1 share the ARTS scientist, so a memorized solution is available to every method, and ARTS still wins 21 of 22 tasks against them. ARTS solutions have TF-IDF similarity 0.05 to the winning solutions against 0.02 for an unrelated one (Appendix B).

7 Discussion

ARTS replaces heuristic parent selection with reasoning over code, logs, scores, and memory; it is the best method on 16 of 22 tasks and improves the mean normalized score by 12.0%, and test-time training raises Qwen3-4B from 0.72 to 1.01 on MLGym.

8 Limitations

We evaluate on ML tasks with a fixed metric and harness; open-ended research, where hypotheses cannot be scored automatically, is outside this setting. Budgets are matched in wall-clock time, not compute (ARTS*: 24 vs. 8 GPU-hours per run), ARTS* is trained per task and not evaluated for transfer, and compaction is the only other long-context treatment we compare against. Compute-matched baselines, cross-task transfer, and other long-context treatments are out of scope and left to future work.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods (LLM coding assistance for the `ARTS` codebase and experiment scripts) and to support qualitative and thematic data analysis (a GPT-5.5 coding agent labeled each proposed hypothesis into one of eight axes for the diversity analysis in §6.2). We have not used generative AI tools to propose or refine the research hypotheses of this paper, to design or provide feedback on research methodology or experiments, to interpret results, to generate synthetic data sets, or to clean and reformat datasets; developing theoretical models or conceptual frameworks, formulating mathematical claims, proving mathematical claims, and translation are not applicable to this work. Additionally, we used generative AI tools to create or edit software code and to edit the research paper to improve readability. We have reviewed all AI-assisted work: LLM-generated code was verified and tested by the authors, and the axis labels were reviewed by the authors alongside the manual labeling of 75 expansions reported in §6.2. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix B lists the models, the audit length R, the number of candidates K, the LoRA and GRPO hyperparameters, the compute per run, and the evaluation-script restoration procedure. Appendix D gives the full scientist and executor prompts, Appendix C the sources of every human-best score, and Appendices F–I the complete search trees behind the qualitative claims. Appendix A reports every ablation with raw scores. We will release the code, prompts, and per-node search trees upon publication.

ETHICS STATEMENT

This work uses public ML benchmarks and involves no human subjects or private data. Agents that automate ML experimentation could concentrate research capability among groups with the most compute and could be misused in unsafe domains; we ran every agent inside a sandbox with a fixed task, restored evaluation scripts, and full logging, and we recommend the same constraints and human oversight for deployment.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. Advances in Neural Information Processing Systems, 2021.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=asgBo3FNdg.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. Nature, 624:570–578, 2023.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. Mlebench: Evaluating machine learning agents on machine learning engineering, 2024. URL https://arxiv.org/abs/2410.07095.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. Mars: Modular agent with reflective search for automated ai research, 2026. URL https://arxiv.org/abs/2602.02660.

Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, Bang Liu, and Chenglin Wu. Sela: Tree-search enhanced llm agents for automated machine learning, 2024. URL https://arxiv.org/abs/2410.17238.

Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report, 2025. URL https://arxiv.org/abs/2412.04604.

Dean De Cock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. Journal of Statistics Education, 19(3), 2011. doi: 10.1080/10691898.2011.11889627. URL https://doi.org/10.1080/10691898.2011.11889627.

Google DeepMind. Gemini 3 flash, 2025. URL https://deepmind.google/models/gemini/flash/.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, Zichen Liang, Boyuan Sun, Tianshuo Peng, Yifan Zhou, Xin Li, Jie Zhou, Liang He, Bo Zhang, and Lei Bai. MLEvolve: A self-evolving framework for automated machine learning algorithm discovery, 2026. URL https://arxiv.org/abs/2606.06473.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai, 2025. URL https://arxiv.org/abs/2411.04872.

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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nature, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Nathan Herr, Tim Rocktäschel, and Roberta Raileanu. Llm-first search: Self-guided exploration of the solution space, 2025. URL https://arxiv.org/abs/2506.05213.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation, 2024. URL https://arxiv.org/abs/2310.03302.

Arnav Kumar Jain, Gonzalo Gonzalez-Pumariega, Wayne Chen, Alexander M Rush, Wenting Zhao, and Sanjiban Choudhury. Multi-turn code generation through single-step rewards. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=aJeLhLcsh0.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. Aide: Ai-driven exploration in the space of code, 2025. URL https://arxiv.org/abs/2502.13138.

Kaggle. Titanic - machine learning from disaster, 2012. URL https://www.kaggle.com/c/titanic. Kaggle competition. Accessed: 2026-05-07.

Kaggle. Histopathologic cancer detection, 2018a. URL https://www.kaggle.com/competitions/histopathologic-cancer-detection. Kaggle competition. Accessed: 2026-05-07.

Kaggle. Toxic comment classification challenge, 2018b. URL https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge. Kaggle competition. Accessed: 2026-05-07.

Kaggle. APTOS 2019 blindness detection, 2019a. URL https://www.kaggle.com/competitions/aptos2019-blindness-detection. Kaggle competition. Accessed: 2026-05-07.

Kaggle. Kuzushiji recognition: Opening the door to a thousand years of Japanese culture, 2019b. URL https://www.kaggle.com/competitions/kuzushiji-recognition. Kaggle competition. Accessed: 2026-05-07.

Kaggle. RSNA-MICCAI brain tumor radiogenomic classification, 2021. URL https://www.kaggle.com/competitions/rsna-miccai-brain-tumor-radiogenomic-classification. Kaggle competition. Accessed: 2026-05-07.

Kaggle. Spaceship titanic, 2022. URL https://www.kaggle.com/competitions/spaceship-titanic. Kaggle competition. Accessed: 2026-05-07.

Kaggle. Vesuvius challenge - ink detection, 2023. URL https://www.kaggle.com/competitions/vesuvius-challenge-ink-detection. Kaggle competition. Accessed: 2026-05-07.

Kaggle. HMS - harmful brain activity classification, 2024. URL https://www.kaggle.com/competitions/hms-harmful-brain-activity-classification. Kaggle competition. Accessed: 2026-05-07.

Andrej Karpathy. autoresearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch, March 2026. GitHub repository.

Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. The automation of science. Science, 324(5923):85–89, 2009.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution, 2025. URL https://arxiv.org/abs/2509.19349.

Zujie Liang, Feng Wei, Wujiang Xu, Lin Chen, Yuxi Qian, and Xinhui Wu. I-MCTS: Enhancing agentic AutoML via introspective monte carlo tree search. In Findings of the Association for Computational Linguistics: EACL 2026, 2026. URL https://arxiv.org/abs/2502.14693.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

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

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. Discoverybench: Towards data-driven discovery with large language models. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=vyflgpwfJW.

Deepak Nathani, Lovish Madaan, Nicholas Roberts, Nikolay Bashlykov, Ajay Menon, Vincent Moens, Mikhail Plekhanov, Amar Budhiraja, Despoina Magka, Vladislav Vorotilov, Gaurav Chaurasia, Dieuwke Hupkes, Ricardo Silveira Cabral, Tatiana Shavrina, Jakob Nicolaus Foerster, Yoram Bachrach, William Yang Wang, and Roberta Raileanu. MLGym: A new framework and benchmark for advancing AI research agents. In Second Conference on Language Modeling, 2025. URL https://openreview.net/forum?id=ryTr83DxRq.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

OpenAI. Introducing openai o3 and o4-mini, Apr 2025. URL https://openai.com/index/introducing-o3-and-o4-mini/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), Advances in Neural Information Processing Systems, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. To backtrack or not to backtrack: When sequential search limits model reasoning, 2025. URL https://arxiv.org/abs/2504.07052.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. Nature, 625:468–475, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent, 2025. URL https://github.com/algorithmicsuperintelligence/openevolve.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Chenglei Si, Zitong Yang, Yejin Choi, Emmanuel Candès, Diyi Yang, and Tatsunori Hashimoto. Towards execution-grounded automated ai research, 2026. URL https://arxiv.org/abs/2601.14525.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In International Conference on Machine Learning, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025. URL https://arxiv.org/abs/2407.04620.

Christopher Sutton, Luca M. Ghiringhelli, Takenori Yamamoto, Yury Lysogorskiy, Lars Blumenthal, Thomas Hammerschmidt, Jacek R. Golebiowski, Xiangyue Liu, Angelo Ziletti, and Matthias Scheffler. Crowd-sourcing materials-science challenges with the NOMAD 2018 Kaggle competition. npj Computational Materials, 5:111, 2019. doi: 10.1038/s41524-019-0239-3. URL https://doi.org/10.1038/s41524-019-0239-3.

Ranjita Thapa, Kai Zhang, Noah Snavely, Serge Belongie, and Awais Khan. The plant pathology challenge 2020 data set to classify foliar disease of apples. Applications in Plant Sciences, 8(9):e11390, 2020. doi: https://doi.org/10.1002/aps3.11390. URL https://bsapubs.onlinelibrary.wiley.com/doi/abs/10.1002/aps3.11390.

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.

Zhaotian Weng, Antonis Antoniades, Deepak Nathani, Zhen Zhang, Xiao Pu, and Xin Eric Wang. Group-evolving agents: Open-ended self-improvement via experience sharing, 2026. URL https://arxiv.org/abs/2602.04837.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Holden Karnofsky, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts, 2025. URL https://arxiv.org/abs/2411.15114.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

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, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), Advances in Neural Information Processing Systems, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL https://arxiv.org/abs/2210.03629.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments, 2019. URL https://arxiv.org/abs/1903.03176.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175.

Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock LLM diversity, 2025. URL https://arxiv.org/abs/2510.01171.

Appendix Contents

A Cost Accounting and Extended Ablations . . . 17
B Additional Experimental Details . . . 23
C Sources for Human-Best Scores . . . 23
D Scientist Prompts . . . 25
E Failure Attribution: Detailed Examples . . . 31
F HMS Search Trajectory: ARTS vs. AIRA . . . 33
G Vesuvius Search Trajectory: ARTS vs. AIRA . . . 34
H MetaMaze Search Trajectory: ARTS vs. AIRA . . . 36
I MountainCar Search Trajectory: ARTS vs. AIRA . . . 37
J Qualitative Analysis of Search Efficiency . . . 39
K Extended Related Work . . . 41

Appendix A gives the full cost accounting and every ablation with raw scores; Appendices B–D give the experimental details, the sources of the human-best scores, and the prompts. Appendix E and the four search-tree figures (§F–§I) back Findings 1–3 in the main text with per-node trajectories, followed by the qualitative analysis and the extended related work.

A Cost Accounting and Extended Ablations

This section reports the full cost accounting and the ablations summarized in §6.4. Cost and normalized scores use the same baseline and human-best normalization as the main results.

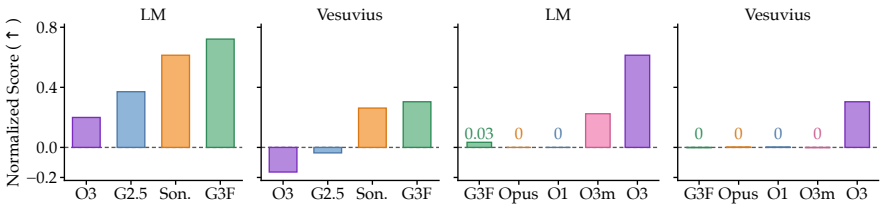
A.1 Cost Accounting

Table 3 reports the average cost of one 8-hour run, including hidden reasoning tokens. We price GPUs at $1.10 per hour for a 40 GB A100, Gemini 3 Flash at $0.50/M input and $3.00/M output tokens, and o3 at $2.00/M input and $8.00/M output tokens. Baselines use one agent for both roles, so all their tokens appear under the executor.

Table 3: Average cost of one run. Tokens and dollar cost per role, GPU-hours, and total. ARTS* hosts its scientist locally, so it has no scientist API cost.

Method	Sci. in	Sci. out	Exec. in	Exec. out	GPU-h	Total
Linear	–	–	3.53M / $1.76	23.1K / $0.07	8	$10.63
AIRA	–	–	2.51M / $1.25	78.6K / $0.24	8	$10.29
MLEvolve	–	–	5.39M / $2.69	729K / $2.19	8	$13.68
ARTS	766K / $1.53	51.7K / $0.41	2.56M / $1.28	96.3K / $0.29	8	$12.31
ARTS*	1.18M / $0	84.2K / $0	2.41M / $1.21	90.5K / $0.27	24	$27.88

A.2 Model and Selection Swaps



(a) (b)

Figure 4: Swap ablations: (a) executor swap with the scientist fixed to o3; (b) scientist swap with the executor fixed to Gemini 3 Flash. Dashed line: normalized score 0.

Table 4: Model swaps. In the executor swap, the scientist is fixed to o3; in the scientist swap, the executor is fixed to Gemini 3 Flash. Each block is a separate single-seed run, so the o3 scientist with Gemini 3 Flash executor rows differ from each other and from Table 1, which reports the mean of three later runs. Normalized scores use the task baseline and human-best values.

Ablation	Task	Model	Raw score	Norm.
Executor	LM (loss ↓)	o3	4.439	0.199
Executor	LM (loss ↓)	Gemini 2.5 Pro	4.238	0.371
Executor	LM (loss ↓)	Claude Sonnet	3.953	0.614
Executor	LM (loss ↓)	Gemini 3 Flash	3.827	0.721
Executor	Vesuvius (F_β ↑)	o3	0.000	-0.164
Executor	Vesuvius (F_β ↑)	Gemini 2.5 Pro	0.091	-0.036
Executor	Vesuvius (F_β ↑)	Claude Sonnet	0.304	0.262
Executor	Vesuvius (F_β ↑)	Gemini 3 Flash	0.334	0.304
Scientist	LM (loss ↓)	Gemini 3 Flash	4.633	0.034
Scientist	LM (loss ↓)	Claude Opus	4.673	0.000
Scientist	LM (loss ↓)	o1	4.673	0.000
Scientist	LM (loss ↓)	o3-mini	4.410	0.224
Scientist	LM (loss ↓)	o3	3.953	0.614
Scientist	Vesuvius (F_β ↑)	Gemini 3 Flash	0.116	-0.001
Scientist	Vesuvius (F_β ↑)	Claude Opus	0.118	0.001
Scientist	Vesuvius (F_β ↑)	o1	0.118	0.001
Scientist	Vesuvius (F_β ↑)	o3-mini	0.116	-0.001
Scientist	Vesuvius (F_β ↑)	o3	0.334	0.304

Table 5: Selection swap. The whole ARTS pipeline (o3 scientist, Gemini 3 Flash executor, audit, verbalized sampling, memory) is kept and reasoning-based parent selection is replaced with UCB; the reasoning rows repeat Table 1.

Ablation	Task	Selection	Raw score	Norm.
Selection	LM (loss ↓)	UCB	4.670	0.003
Selection	LM (loss ↓)	Reasoning (ARTS)	3.827	0.721
Selection	Battle of Sexes (payoff ↑)	UCB	1.550	0.818
Selection	Battle of Sexes (payoff ↑)	Reasoning (ARTS)	2.000	1.517
Selection	Blotto (score ↑)	UCB	0.034	0.377
Selection	Blotto (score ↑)	Reasoning (ARTS)	0.249	0.664
Selection	MountainCar (reward ↑)	UCB	65.70	0.489
Selection	MountainCar (reward ↑)	Reasoning (ARTS)	95.73	0.950
Selection	Breakout (reward ↑)	UCB	67.10	0.357
Selection	Breakout (reward ↑)	Reasoning (ARTS)	78.00	0.570
Selection	Meta Maze (reward ↑)	UCB	38.97	0.632
Selection	Meta Maze (reward ↑)	Reasoning (ARTS)	51.20	0.965
Selection	Vesuvius (F_β ↑)	UCB	0.231	0.278
Selection	Vesuvius (F_β ↑)	Reasoning (ARTS)	0.551	0.663
Selection	HMS (KL ↓)	UCB	0.930	0.447
Selection	HMS (KL ↓)	Reasoning (ARTS)	0.499	0.809
Selection	Titanic (acc ↑)	UCB	0.962	3.063
Selection	Titanic (acc ↑)	Reasoning (ARTS)	0.984	3.406
Selection	CIFAR-10 (acc ↑)	UCB	0.965	0.942
Selection	CIFAR-10 (acc ↑)	Reasoning (ARTS)	0.971	0.954
Selection	Fashion MNIST (acc ↑)	UCB	0.952	0.867
Selection	Fashion MNIST (acc ↑)	Reasoning (ARTS)	0.958	0.917
Selection	House Price (R^2 ↑)	UCB	0.940	0.545
Selection	House Price (R^2 ↑)	Reasoning (ARTS)	0.939	0.536
Selection	MNLI (acc ↑)	UCB	84.40	0.797
Selection	MNLI (acc ↑)	Reasoning (ARTS)	84.71	0.805
Selection	Prisoner's Dilemma (payoff ↑)	UCB	2.603	0.368
Selection	Prisoner's Dilemma (payoff ↑)	Reasoning (ARTS)	2.857	0.772
Selection	Spaceship Titanic (acc ↑)	UCB	0.833	1.006
Selection	Spaceship Titanic (acc ↑)	Reasoning (ARTS)	0.836	1.010
Selection	Nomad 2018 (RMSLE ↓)	UCB	0.064	0.986
Selection	Nomad 2018 (RMSLE ↓)	Reasoning (ARTS)	0.064	0.986
Selection	Jigsaw Toxic (AUC ↑)	UCB	0.980	0.982
Selection	Jigsaw Toxic (AUC ↑)	Reasoning (ARTS)	0.980	0.982
Selection	APTOS 2019 (QWK ↑)	UCB	0.921	0.984
Selection	APTOS 2019 (QWK ↑)	Reasoning (ARTS)	0.930	0.994
Selection	Plant Pathology (AUC ↑)	UCB	0.994	1.021
Selection	Plant Pathology (AUC ↑)	Reasoning (ARTS)	0.995	1.023
Selection	Histopathologic Cancer (AUC ↑)	UCB	0.993	0.986
Selection	Histopathologic Cancer (AUC ↑)	Reasoning (ARTS)	0.995	0.990
Selection	Kuzushiji (F1 ↑)	UCB	0.851	0.896
Selection	Kuzushiji (F1 ↑)	Reasoning (ARTS)	0.843	0.887
Selection	RSNA Brain Tumor (AUC ↑)	UCB	0.655	1.281
Selection	RSNA Brain Tumor (AUC ↑)	Reasoning (ARTS)	0.673	1.430

A.3 Token Usage and Components

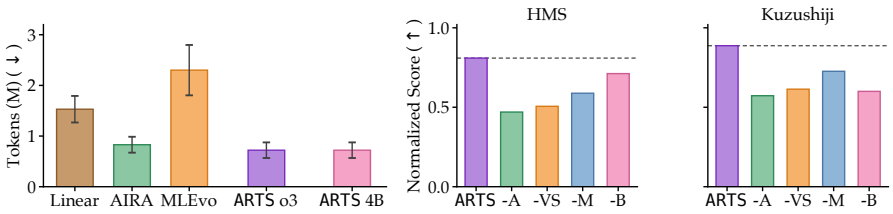


Figure 5: Token and component ablations. (a) Token usage estimated from saved trajectory text. (b) Component ablations, where A, VS, M, and B denote audit, verbalized sampling, memory, and initial breadth. The dashed line marks the full ARTS score.

Table 6: Estimated token and executor-call usage on the 11 MLGym tasks with recovered event counts. Tokens are estimated from saved trajectory text using characters divided by four.

Method	Tokens (M)	SE (M)	Runs	Calls
Linear	1.53	0.26	26	417
AIRA	0.83	0.16	24	244
MLEvolve	2.30	0.50	33	605
ARTS (o3)	0.72	0.15	26	209
ARTS (Qwen)	0.72	0.15	26	209

Table 7: Component ablations for ARTS. Normalized scores use the same baseline and human-best normalization as the main results.

Task	Variant	Removed component	Raw score	Norm.
HMS (KL ↓)	ARTS	None	0.499	0.809
HMS (KL ↓)	A	Audit	0.903	0.470
HMS (KL ↓)	B	Verbalized sampling	0.860	0.506
HMS (KL ↓)	C	Memory	0.762	0.588
HMS (KL ↓)	D	Initial breadth	0.615	0.712
Kuzushiji (F1 ↑)	ARTS	None	0.843	0.887
Kuzushiji (F1 ↑)	A	Audit	0.544	0.573
Kuzushiji (F1 ↑)	B	Verbalized sampling	0.583	0.614
Kuzushiji (F1 ↑)	C	Memory	0.690	0.726
Kuzushiji (F1 ↑)	D	Initial breadth	0.570	0.600

A.4 TTT Reward and Episode Structure

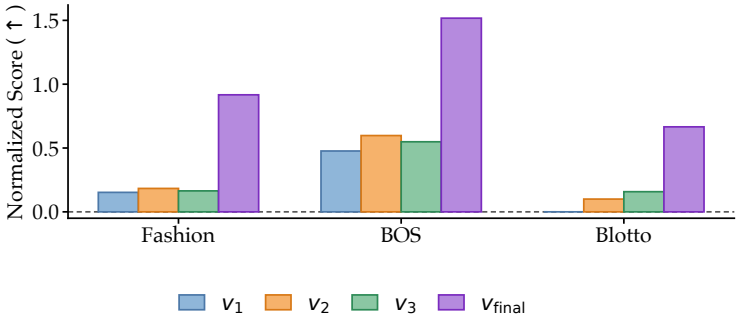


Figure 6: Reward-function ablation for test-time training. Scores are normalized by the task baseline and human-best value.

Table 8: Reward-function ablation. Each task entry reports raw score followed by normalized score in parentheses.

Version	Reward	Fashion MNIST	BOS	Blotto	Mean norm.
v_1	$\mathbf{1}[s_t > s_{\text{base}}]$	0.866 (0.152)	1.330 (0.477)	-0.248 (0.000)	0.210
v_2	$\{-0.5, 0, 0.2, 1\}$ with $+1$ if $s_t \geq 0.80$	0.870 (0.183)	1.408 (0.598)	-0.173 (0.100)	0.294
v_3	$\mathbf{1}[s_t > s_t^\star]$	0.868 (0.164)	1.376 (0.549)	-0.130 (0.158)	0.290
v_{final}	$\{-0.5, -0.2, 0, 1\}$ with $+1$ if $s_t \geq s_{70}(\mathcal{G}_t)$	0.958 (0.917)	2.000 (1.517)	0.250 (0.666)	1.033

Table 9: Episode-structure ablation. Each entry reports raw score followed by normalized score in parentheses.

Task	Tree-per-episode	Single-step GRPO
Fashion MNIST	0.853 (0.04)	0.948 (0.83)
BOS	1.408 (0.60)	1.442 (0.65)
Blotto	-0.173 (0.10)	0.344 (0.79)
Titanic	0.947 (2.83)	1.000 (3.66)
Mean norm.	0.89	1.48

A.5 Diversity After TTT

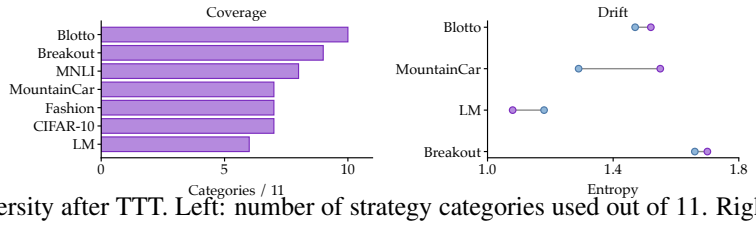


Figure 7: Diversity after TTT. Left: number of strategy categories used out of 11. Right: early (blue) and late (purple) entropy across rollouts.

Table 10: Diversity after TTT. Categories are assigned by keyword matching over proposed strategy text. Entropy is reported for tasks with early/late rollout statistics.

Task	Top categories by share	Used / 11	Early H	Late H
Blotto	Other 37%, feature_eng 36%, data_aug 8%	10	1.47	1.52
Breakout	rl_specific 38%, CNN 19%, MLP 18%	9	1.66	1.70
MNLI	hyperparam 22%, data_aug 21%, rl_specific 19%	8	—	—
MountainCar	feature_eng 47%, MLP 22%, rl_specific 15%	7	1.29	1.55
Fashion MNIST	data_aug 57%, CNN 21%, hyperparam 11%	7	—	—
CIFAR-10	data_aug 55%, hyperparam 30%, CNN 5%	7	—	—
LM	Other 65%, feature_eng 12%, data_aug 10%	6	1.18	1.08

A.6 Hypothesis Selection, Audit Budget, and Reward Window

Table 11 compares rules for choosing one of the K verbalized candidates on Blotto. Uniform sampling, which ignores the verbalized probabilities, is best. Rules that trust the probabilities, either by taking the most probable candidate or by sampling in proportion to them, fall below the baseline.

Table 11: Selection rule for verbalized sampling on Blotto (mean best score).

Selection rule	Mean best
Uniform (ours)	**0.251**
First listed candidate	0.000
Most probable candidate	−0.247
Temperature sampling	−0.064
Weighted by verbalized probability	−0.244

Table 12 sweeps the audit budget R. On Vesuvius and Prisoner's Dilemma the score rises up to $R = 3$ or $R = 4$ and drops at $R = 5$, where the audit consumes budget that expansion would otherwise use. We use $R = 3$ in all experiments.

Table 12: Audit budget R (number of read-only scientist calls before the first expansion).

Task	$R = 0$	1	2	3	4	5
Vesuvius ($F_{0.5}$ ↑)	0.380	0.460	0.460	0.551	**0.566**	0.524
Prisoner's Dilemma (payoff ↑)	2.510	2.608	2.810	**2.850**	2.599	2.409
Blotto (score ↑)	−0.241	0.247	0.249	**0.251**	0.250	0.248

Table 13 sweeps the window of recent scores over which the percentile threshold s_{70} of Eq. 2 is computed, for ARTS* on Blotto.

Table 13: Window size for the percentile reward, ARTS* on Blotto (best score).

Window	8	16	32	64	128	256
Blotto	0.247	0.240	0.251	**0.344**	0.305	0.250

A.7 Comparison with TTT-Discover

We compared this reward with the continuous entropic objective of TTT-Discover (Yuksekgonul et al., 2026) on Language Modeling. The entropic objective reaches a loss of 4.04, against 3.52 with the percentile reward, and its rewards have variance 0.07 against 0.26, which gives GRPO weaker advantages. Two further differences separate ARTS* from TTT-Discover: TTT-Discover selects the state to expand with PUCT, while ARTS* trains the parent choice, and TTT-Discover conditions on a sampled state and its parent, while ARTS* conditions on the whole tree with memory and tool calls.

B Additional Experimental Details

Execution environment. All runs execute inside Apptainer containers with the same executor sandbox, action set, and validation interface. We restore the evaluation script before every validation. We added this guard because agents sometimes attempted to modify evaluation code despite instructions not to do so. Restoring the script prevents metric hacking and keeps the comparison focused on legitimate changes to the submission or training code.

Models and prompts. For the main `ARTS` experiments, the scientist is OpenAI o3 (OpenAI, 2025) and the executor is Gemini 3 Flash (DeepMind, 2025). Model versions: `o3-2025-04-16` (OpenAI API), `gemini-3-flash-preview` (Gemini API), and `Qwen3-4B-Instruct-2507` (Hugging Face). For baselines that do not separate scientist and executor roles, we use Gemini 3 Flash for both reasoning and coding. We also report ablations that swap the scientist and executor models. The scientist and executor prompts are given in Appendix D.

`ARTS` settings. Each run receives an 8-hour wall-clock budget. `ARTS` uses an audit length $R = 3$ scientist calls. During proposal, the scientist enumerates $K = 5$ candidate hypotheses using verbalized sampling, as described in §4.1. One candidate is sampled and passed to the executor, which implements and validates the experiment in the sandbox.

TTT settings. For test-time training, we use Qwen3-4B-Instruct (Yang et al., 2025) as the scientist. TTT uses LoRA adapters of rank 32 with $\alpha = 64$ and dropout 0 on all attention and MLP linear projections. We train with AdamW using learning rate 1×10^{-5}, constant schedule, $\beta = (0.9, 0.999)$, weight decay 0.01, and gradient clipping at 1.0. Each GRPO step samples $N = 8$ rollouts per group and applies one inner update with KL penalty $\beta_{\mathrm{KL}} = 0.01$ against the frozen pretrained base. Per-token importance ratios are clipped to $[0.8, 1.28]$.

Rollout limits. Rollouts use sequence length 8192 with up to 120 scientist turns of 1024 tokens each. Each sampled candidate is executed once, with no best-of-N reranking. LoRA adapters are checkpointed and broadcast to the inference server every 5 GRPO steps. Training is capped at 100 GRPO steps or the 8-hour wall-clock budget, whichever comes first.

Compute. All experiments use 40 GB A100 GPUs. Each inference-only experiment, including Linear, AIRA, MLEvolve, and `ARTS` without TTT, uses one GPU for the executor sandbox, for 8 GPU-hours per run. Each TTT experiment uses three GPUs, one for the LoRA trainer, one for the vLLM inference server hosting Qwen3-4B, and one for the executor sandbox, for 24 GPU-hours per run.

Memory entries. After every expansion the scientist writes one sentence under the keyword `MEMORY:`. The prompt tells it not to repeat anything already in memory and gives examples of entries to include and exclude. An entry from a Vesuvius run reads: "The 0.0369 score from `root_0` confirms the pipeline is functional but the 3-difference-slice input representation discards $\sim$95% of depth information; the model is blind to the 3D context that distinguishes ink from papyrus, and no loss, optimizer, or regularization change can recover signal that was never fed in." Two further entries, from HMS and MountainCar: "EfficientNet-B0 on the 3-channel log-mel input regresses to KL 0.702; the backbone swap is not the bottleneck, the spectrogram front end is." "Reward stays at -200 for every episode when the policy is trained from raw position and velocity; the sparse reward is never reached, so shaping the reward by potential energy is the first thing to try."

Contamination. The ten MLEBench competitions were released before the training cutoffs of o3 and Gemini 3 Flash, so public solutions may appear in their training data. Two observations bound this effect. First, all code in every method is written by the same executor model, so a memorized solution is equally available to every baseline. Second, `ARTS` solutions differ from the public winning solutions: their TF-IDF cosine similarity is 0.05 on average (max 0.09), against 0.02 for the solution to an unrelated competition. For example, on APTOS `ARTS` trains a single ConvNeXt-small, while the winning entries ensemble EfficientNet-B5/B7 models with external data.

C Sources for Human-Best Scores

Table 14 records the provenance of the human-best values reported in Table 1. For MLEBench tasks, the value is the top-1 Kaggle leaderboard score for the corresponding competition. For MLGym tasks,

the value is the published best-known result for the benchmark task or environment. We include the table to make the normalization constants auditable.

Table 14: Sources for the human-best scores used to normalize results. MLGym values follow the published benchmark task definitions. MLEBench values use the public Kaggle leaderboard protocol from MLE-bench, with task-level sources shown for auditability.

Task	Human Best	Source
Titanic	0.830	MLGym Nathani et al. (2025); Kaggle task Kaggle (2012)
CIFAR-10	0.994	MLGym Nathani et al. (2025); CIFAR-10 Krizhevsky (2009)
Fashion MNIST	0.968	MLGym Nathani et al. (2025); Fashion-MNIST Xiao et al. (2017)
House Price	0.990	MLGym Nathani et al. (2025); Ames Housing De Cock (2011)
MNLI	92.50	MLGym Nathani et al. (2025); MultiNLI
Lang. Modeling	3.500	MLGym Nathani et al. (2025); FineWeb Penedo et al. (2024)
Battle of Sexes	1.667	MLGym task specification Nathani et al. (2025)
Prisoner's Dilemma	3.000	MLGym task specification Nathani et al. (2025)
Blotto	0.500	MLGym task specification Nathani et al. (2025)
MountainCar	99.00	MLGym Nathani et al. (2025); OpenAI Gym Brockman et al. (2016)
Breakout	100.00	MLGym Nathani et al. (2025); MinAtar Young & Tian (2019)
Meta Maze	52.50	MLGym task specification Nathani et al. (2025)
Spaceship Titanic	0.828	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2022)
Nomad 2018	0.051	MLE-bench/Kaggle leaderboard Chan et al. (2024); Sutton et al. (2019)
Jigsaw Toxic	0.989	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2018b)
APTOS 2019	0.936	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2019a)
Plant Pathology	0.984	MLE-bench/Kaggle leaderboard Chan et al. (2024); Thapa et al. (2020)
Histopathologic Cancer	1.000	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2018a)
Vesuvius Ink Detection	0.831	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2023)
Kuzushiji Recognition	0.950	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2019b)
HMS Brain Activity	0.272	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2024)
RSNA Brain Tumor	0.621	MLE-bench/Kaggle leaderboard Chan et al. (2024); Kaggle (2021)

Titanic's full labels are public and every method exceeds its leaderboard best, so we treat it as a sanity check and exclude it when counting tasks that exceed the human-best reference.

Metrics: accuracy (Titanic, CIFAR-10, F-MNIST, MNLI, Spaceship), R^2 (House Price), loss (Lang. Model.), payoff (Battle of Sexes, Prisoner's D.), score (Blotto), reward (MountainCar, Breakout, Meta Maze), MCW-RMSLE (Nomad), CW-AUC (Jigsaw), QWK (APTOS), MCW-AUC (Plant Path.), AUC (Histopath., RSNA), $F_{0.5}$ (Vesuvius), F1 (Kuzushiji), KL (HMS Brain).

D SCIENTIST PROMPTS

The scientist operates in two turns per node expansion (three during the research phase). Turn 1 is shared between the active and research phases; Turn 2 differs. All `{placeholders}` are filled at runtime.

D.1 TURN 1 — INSPECTION

Scientist – Turn 1: Inspection

```
You are an ML research ADVISOR. You propose ONE experiment per turn.
A separate coder (the "executor") executes it. You must NEVER write
code (no ``` blocks, no import statements, no def/class).

Think like a scientist: analyze what has been tried, identify gaps,
form a hypothesis, and propose a direction.

Be aware that per-node training cost varies by task -- some nodes
finish in minutes, others take 30+ minutes. Coupled changes (LR
schedule + longer training, augs + the regularization that enables
them) bundle naturally; unrelated levers do not.

## Task
{task_description}

## Task Details (this is what the executor sees)
{task_details}

The metric is: {metric_name} ({direction} is better)
Baseline score (no model, just default): {baseline_score}

## How This Works

Each time you propose an experiment, the executor writes code from
scratch in a container, runs it, and validates. It has {max_actions}
actions (shell commands) per attempt. Each attempt creates one "node"
in your search tree. The executor can run any single command for up
to 45 minutes.

## Budget

Total run time: {time_budget_min} min.
Time elapsed: {time_elapsed_min} min.
Time remaining: {time_remaining_min} min.
Nodes scored so far: {nodes_done}
  (avg {avg_per_node_min:.1f} min each).
At the current per-node pace, ~{nodes_remaining_est} more nodes fit
in the remaining budget. Factor this into how ambitious each
candidate is.

IMPORTANT: The executor already has ALL source files from the
workspace pre-loaded in its context. Do NOT waste a node asking it
to "read" or "examine" files -- every experiment should be an
ACTIONABLE change (modify config, swap architecture, tune
hyperparameters), never exploration.

INTEGRITY: Do NOT modify evaluation files, opponent strategy files
(e.g. target.py), or any read-only starter code. Do NOT use
monkey-patching, sys.modules hacking, or any technique to manipulate
the scoring system. All such modifications are reverted before
evaluation. Only modify YOUR submission files (e.g. strategy.py,
baseline.py, train_and_predict.py). Legitimate improvements only.

## Your Search Tree
{tree_view}

## Your Accumulated Knowledge
{memory_section}

## Your Task Now
```

```
Before making a decision, you have two tools:

1. INSPECT nodes -- see the actual commands and output the executor
   ran for any node. Lets you understand EXACTLY what was tried and
   why it succeeded or failed.

2. READ files -- see the contents of any workspace file (e.g.
   target.py, evaluate.py, baseline.py, strategy.py). Lets you
   understand the task code, opponent strategy, evaluation logic, or
   data format before proposing an experiment.

Respond in EXACTLY this format:

INSPECT: node_id_1, node_id_2
[OR]
INSPECT: NONE

READ: filename1.py, filename2.py
[OR]
READ: NONE

Brief explanation of what you want to understand.
```

D.2 TURN 2 — DECISION (ACTIVE SEARCH)

Scientist – Turn 2: Decision

```
Good. Now make your decision.

{code_inspection}

You are an ML research ADVISOR. You propose ONE experiment per turn.
You must NEVER write code (no ``` blocks, no import statements, no
def/class).

Think like a scientist: analyze what has been tried, identify gaps,
form a hypothesis, then propose a specific experiment.

Tree stats: {explore_stats}
Time remaining: {time_remaining_min} min
  ({nodes_done} scored, ~{avg_per_node_min:.1f} min/node).

## Rules

- NO CODE. Describe everything in precise English.
- DIVERSITY: If one axis dominates (>50% of attempts), deliberately
  explore an UNEXPLORED axis. Axes: architecture, loss,
  data_representation, data_augmentation, regularization, optimizer,
  hp, combination. Note: data_representation (what ENTERS the model
  -- input channels, slice ranges, resolution, patch size) and
  data_augmentation (transforms applied to each sample -- flips,
  rotations, mixup, etc.) are TWO DIFFERENT axes. Be specific --
  name exact technique names and parameter values.
- BASELINE AUDIT (mandatory before any loss/optimizer/aug/reg tweak):
  audit the data pipeline. What signal does the raw input contain,
  and what fraction does the current pipeline preserve? Identify the
  single largest information-loss step. If a downstream tweak cannot
  plausibly close a gap the pipeline itself creates, fix the pipeline
  FIRST. State the bottleneck in your ANALYSIS.
- EVOLVE: Two valid ways to build on prior nodes:
  (a) LAYER -- extend ONE high-scoring parent with a new idea. Set
      PARENT to that parent, COMBINES: NONE.
  (b) MERGE -- combine high-scoring nodes from DIFFERENT branches
      that succeeded on DIFFERENT axes. Set PARENT to the stronger;
      list others in COMBINES. Reserve MERGE for when the tree has
      >=2 distinct branches with non-trivial scores and 3+ recent
      single-parent attempts have stalled.
- BUILD ON SIGNAL: Early in a task, prefer single-axis probes. Once
  axes are individually validated, compound them.
- Failed nodes are DATA -- analyze WHY they failed. A code failure
  does NOT mean the approach is wrong.
- REGRESSION INVESTIGATION: if any recently completed node scored
  >20% below its parent, you MUST have INSPECTed it and emit a
  CLASSIFICATION: line. Valid values:
    IDEA-WRONG: <node_id> -- <brief why, from inspection>
    IMPLEMENTATION-WRONG: <node_id> -- <specific code bug>
    NONE (only if no regression >20% exists)
  If IMPLEMENTATION-WRONG: your EXPERIMENT must target the SAME
  AXIS with corrected executor instructions.
- Do NOT abandon promising nodes prematurely (< 3 children).
- If 3+ refinements in the same family stalled, switch families
  (classical ML -> deep learning, CNN -> transformer, etc.).

## Your Output

Respond in EXACTLY this format:
```

```
ANALYSIS:
[Thorough analysis:
 - What has been tried and what scores?
 - What axes are MISSING or underexplored?
 - BASELINE AUDIT: largest information-loss step?
 - Which high-scoring nodes could improve further?
 - What do the failures tell us?]

Enumerate {n_candidates} fundamentally different candidates. Span
DIFFERENT axes OR DIFFERENT families within an axis.
{cap_instruction}

The system samples ONE candidate weighted by <probability>. Be
honest -- DO NOT inflate your favorite. The system, not you, picks.

<candidates>
<response>
<direction>[1 sentence naming the family/technique]</direction>
<probability>NUMBER</probability>
<plan>
HYPOTHESIS: [1-2 sentences: why this improves performance]
EXPERIMENT: [3-6 sentences: WHAT (exact models, values), HOW
  (which components to swap), WHY (expected outcome). As precise
  as a methods section. No code.]
PARENT: [node_id to build on, or "root"]
COMBINES: [node_ids to merge, or NONE]
AXIS: [architecture | loss | data_representation |
       data_augmentation | regularization | optimizer |
       hp | combination]
MODE: [explore | exploit]
</plan>
</response>
[...exactly {n_candidates} responses; probs sum to 1.0...]
</candidates>

MEMORY:
[One sentence: what you LEARNED. Must include evidence (score,
error) and an insight. Do NOT repeat prior memory.
GOOD: "CatBoost (0.91) + LightGBM (0.90) plateau -- try
      feature engineering next."
BAD:  "CatBoost works well." (no new insight)
Write NONE if no genuinely new insight.]

CLASSIFICATION:
[Mandatory. Exactly ONE of:
  IDEA-WRONG: <node_id> -- <brief reason from inspection>
  IMPLEMENTATION-WRONG: <node_id> -- <specific bug>
  NONE
If IMPLEMENTATION-WRONG, EXPERIMENT above must target the SAME
AXIS as the regressed node with corrected instructions.]
```

D.3 TURN 2 — RESEARCH PHASE

Scientist – Research Phase Turn 2

```
You are in the RESEARCH PHASE -- NOT proposing experiments yet.

{code_inspection}

Your job is to BUILD A MENTAL MODEL of the task before running
experiments. A good researcher spends the first day understanding
the problem: what the data looks like, what the metric rewards, what
the baseline computes, where the easy wins are, where the hidden
traps are. That is what this turn is for.

Do NOT propose an experiment. Do NOT pick a node to expand. Write
structured findings that future-you will use to guide the real
search.

## Your Output

Respond in EXACTLY this format:

FINDINGS:
[3-8 specific, concrete, actionable observations. Examples:
 - "Input volumes have 65 z-slices but the baseline uses only
    slices 28-33 -- ~90% of z-axis signal is discarded."
 - "Evaluation uses F0.5 which weights precision 2x recall --
    threshold choice will matter a lot."
 - "Fragment 1 has ~20x more positive pixels than Fragment 2 --
    class imbalance differs across training fragments."
No proposals. Findings only.]

OPEN QUESTIONS:
[2-4 concrete things you still don't understand and would
investigate via READ / INSPECT in future research turns.
Be specific -- not "how does training work" but "what exactly
does the eval script consider a valid submission format?"]

MEMORY:
[1-3 one-sentence insights for the active search phase. Each must
be novel, evidence-backed, and action-implying.
GOOD: "Baseline averages 5 central z-slices then triples them --
       ~90% of z-axis info discarded; any downstream tweak
       ceiling-limits around this loss."
BAD:  "The task is hard." (too vague, no action implied)
Write NONE if no genuinely new insight.]
```

E FAILURE ATTRIBUTION: DETAILED EXAMPLES

This appendix expands the main-text Finding 1 with full per-node trajectories. Each example contrasts what a score-only policy would read from the regression with what ARTS's idea-wrong / implementation-wrong attribution actually inferred, and what the next expansion did as a consequence.

E.1 HMS BRAIN ACTIVITY: FULL TRAJECTORY

HMS Harmful Brain Activity Classification (MLE-bench; KL divergence, lower is better; uniform-prediction baseline KL 1.462). The trajectory below is from the ARTS run `llmg_C_vs_C_v3` reported in Table 1. The full tree with scientist tool calls is in Fig. 8 (Appendix F).

Initial drafts (depth 1). ARTS drafts five families in parallel. The strongest is a pretrained ResNet-50 over 3-channel log-mel spectrograms (`root_2`, KL 0.557); a 1-channel ResNet-18 baseline (`root_0`, 0.623) and a data-axis variant (`root_1`, 0.562) are sibling families.

Two consecutive regressions (depth 2). The two children of the strong ResNet-50 family both regress: `root_2_0` swaps the backbone to EfficientNet-B0 and lands at KL 0.702 (a +0.145 regression); `root_2_1` keeps ResNet-50 and adds SpecAugment (0–2 frequency masks $\leq$ 12 bins, 0–2 time masks $\leq 20\%$ of time steps) and lands at KL 0.565 (a small +0.008 regression). A score-only policy reads two regressions in a row as evidence that the spectrogram-CNN family is exhausted.

Failure-attribution decision. ARTS re-reads the training logs of both regressed nodes. The validation KL is still decreasing at the last epoch in both runs — SpecAugment in particular makes the optimisation slower because it injects augmentation noise. The scientist classifies the regression as **implementation-wrong** (under-trained under heavy augmentation, not a failure of the spectrogram representation) and retains the ResNet-50 + spectrogram family.

Recovery and refinement (depth 3+). The next expansions stay on the ResNet-50 + spectrogram axis and improve KL monotonically: `root_2_2` adds Mixup with Beta$(0.4, 0.4)$ to ResNet-50 (KL 0.527, the first improvement over the parent); `root_2_2_2` stacks Mixup and SpecAugment together (0.519); `root_2_2_2_0` adds per-channel normalisation with dataset-level statistics (μ, σ computed across the 10,024 training spectrograms: $\mu_{C_0} = 1.10$, $\mu_{C_1} = 0.74$, μ_{C_2} similar) to reach 0.516. Continued refinement on the same family ultimately reaches **KL 0.467** — the best on the task.

Contrast with AIRA on the same task. AIRA's first draft also finds a strong spectrogram-CNN family at KL 0.513. Its next two expansions, however, pivot to entirely different pipelines: a hand-crafted feature pipeline using `scipy.signal` with `skew`/`kurtosis` statistics, and a derivative-stacked EfficientNet over the same spectrograms. The children of these pivots diverge catastrophically — `root_2_0` reaches KL 4.08 (roughly 3× worse than baseline) and `root_2_0_0` reaches KL 9.43 — and AIRA never returns to the original 0.513 family. The cost of conflating "why" with "that" is visible here: a working family abandoned after one node is a working family the search never refines.

E.2 APTOS DIABETIC RETINOPATHY: PRETRAINED-FEATURE COLLAPSE

APTOS 2019 (MLE-bench; quadratic-weighted kappa, higher is better).

The collapsed first attempt. The initial draft is a ConvNeXt-Base backbone, fully fine-tuned end-to-end at learning rate 10^{-3}. The training loss decreases over the first epoch but the held-out QWK is exactly **0.000**: the high learning rate erases the pretrained ImageNet features within the first epoch and the model converges to a constant prediction.

Score-only attribution. A score-only policy reads the 0.000 as evidence that ConvNeXt is the wrong backbone for this task and pivots to a different architecture family (typical baseline responses are ResNet-50, EfficientNet, or a custom shallow CNN).

ARTS's attribution. The scientist inspects the training log, observes that the train loss decreases monotonically while the validation QWK is pinned at 0.000, and reads this as collapse to a degenerate

constant prediction — consistent with the learning rate being too large to preserve the pretrained features. The regression is classified as **implementation-wrong (learning rate schedule)** and the ConvNeXt-Base backbone is retained.

Corrected execution. The next expansion edits the training script to: (i) train the classification head only for one epoch with the backbone frozen (linear-probe warmup); then (ii) unfreeze the backbone and continue at learning rate 3×10^{-5} with the same optimiser. The corrected run reaches QWK $\mathbf{0.92}$.

E.3 Vesuvius Ink Detection: a broken validation split

Vesuvius Ink Detection (MLE-bench; $F_{0.5}$, higher is better).

A misleadingly low score. The first DeepLabV3-ResNet50 draft uses a 3-channel input ($z-1, z, z+1$) and reaches $F_{0.5} = 0.151$. A subsequent expansion widens the input to 11 channels ($z-5$ to $z+5$, intended to capture more z-axis context) and lands at $F_{0.5} = 0.165$ — barely above the 3-slice version.

Score-only attribution. A score-only policy reads the 0.165 as evidence that the wider receptive field does not help, prunes the wide-input branch, and pivots to a different architecture (this is what AIRA does on the same task — see Fig. 9 — chaining 3D U-Net, 2.5D U-Net, CRNN, PointNet, sparse-3D conv, Video-ViT, and Swin-UNETR down a single chain, each abandoned after one node).

`ARTS`'s attribution. The scientist re-reads the training log line by line. The log contains the line `DATA: train=4000 val=0`: the validation split was set to zero by the training script, so `val_metric` is constant at zero throughout training and the reported $F_{0.5} = 0.165$ is computed on an unrepresentative held-out shard. The regression is classified as **implementation-wrong (broken validation split)** rather than as a failure of the wide-input idea.

Corrected execution. The next expansion retains the DeepLabV3 + wide-slice family with the validation split repaired, then adds the second scroll fragment to the training data (`root_3`, $F_{0.5} = 0.479$ — a large jump), and finally adds tile/flip augmentation on top of fragment-2 (`root_3_1`, $F_{0.5} = \mathbf{0.575}$), the best on the task.

E.4 Why the same pattern shows up across tasks

The three examples are not isolated. They share a common structure: a hypothesis that is sound as a research direction surfaces a low score because of an execution-side artefact (an over-aggressive learning rate, an under-trained model under augmentation, a broken validation split). A score-only policy reads the low score as evidence against the hypothesis and pivots; the working family is then either lost entirely (HMS, Vesuvius under AIRA) or rediscovered many nodes later in a degraded form (Vesuvius: AIRA only returns to the 2.5D-pretrained variant at depth 13, $F_{0.5} = 0.510$). `ARTS`'s requirement to label every regression as idea-wrong or implementation-wrong forces the scientist to inspect the code and the log before reassigning the budget, which in each case here points to a specific, fixable execution issue rather than to abandonment.

F HMS SEARCH TRAJECTORY: ARTS VERSUS AIRA

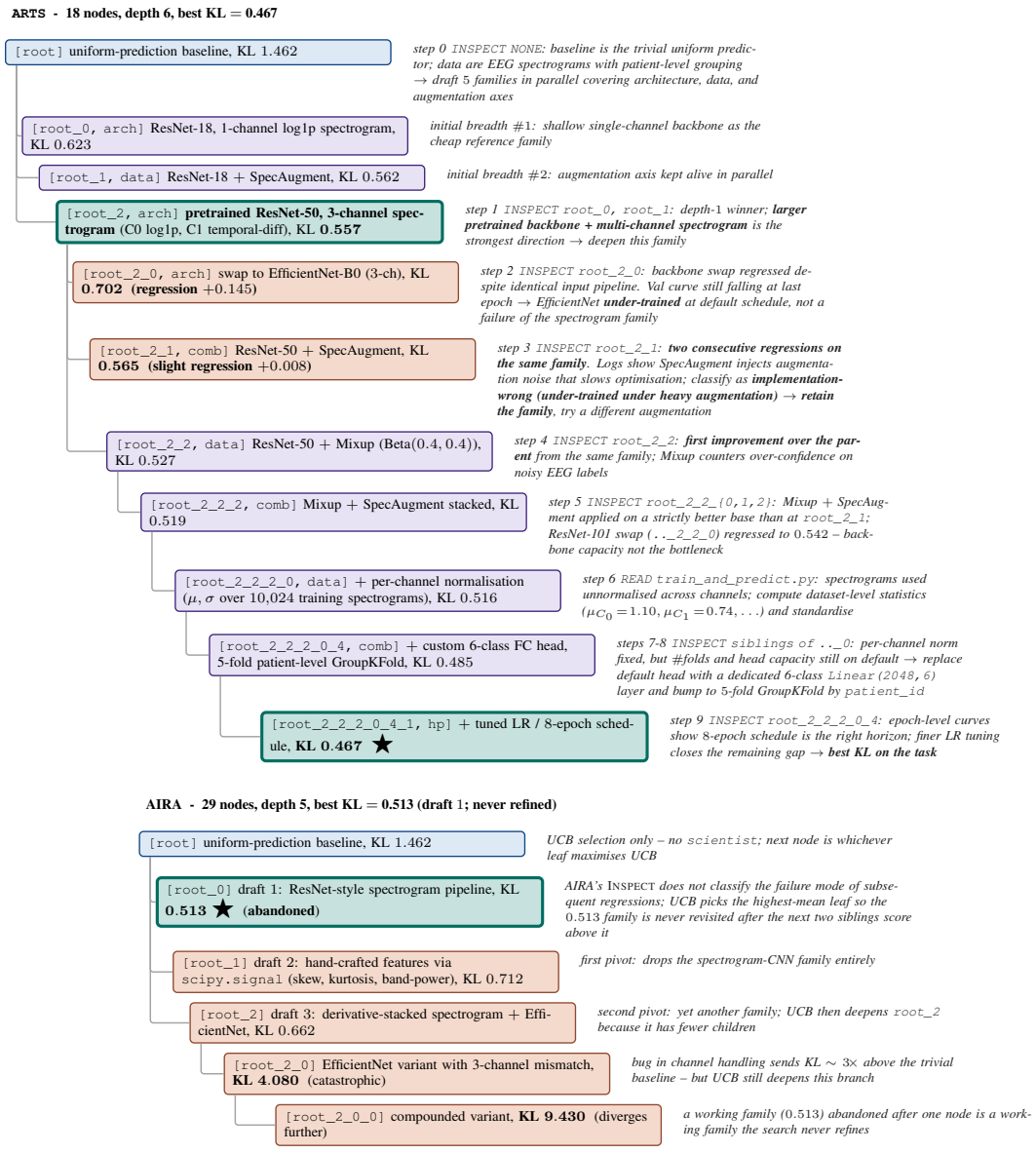


Figure 8: HMS Brain Activity search trees. ARTS (top) drafts a pretrained ResNet-50 over 3-channel log-mel spectrograms at depth 1 and reaches KL **0.557**. The next two children regress (EfficientNet-B0, 0.702; SpecAugment, 0.565). ARTS re-reads the regressed nodes' training logs, classifies them as implementation-wrong (under-trained under heavy augmentation), retains the ResNet-50 + spectrogram family, and the next several expansions compound Mixup, SpecAugment, per-channel normalisation, a 5-fold patient-level GroupKFold, and an LR / schedule sweep to reach **KL 0.467** – the best on the task. AIRA (bottom) reaches a comparable family at its first draft (0.513) but its next two drafts pivot to hand-crafted features and a derivative-stacked EfficientNet; UCB then deepens the latter pivot whose children diverge catastrophically to KL **4.08** and **9.43**, and AIRA never returns to the original 0.513 family.

We contrast ARTS and AIRA on HMS Harmful Brain Activity Classification (MLE-bench; KL divergence, lower is better; uniform-prediction baseline KL 1.462). Tree nodes are coloured **blue** (baseline), **purple** (progress), **teal** (new best, ★), and **red** (regression / abandoned family). Italic margin notes give the scientist's tool call (INSPECT/READ) at that step together with the reasoning

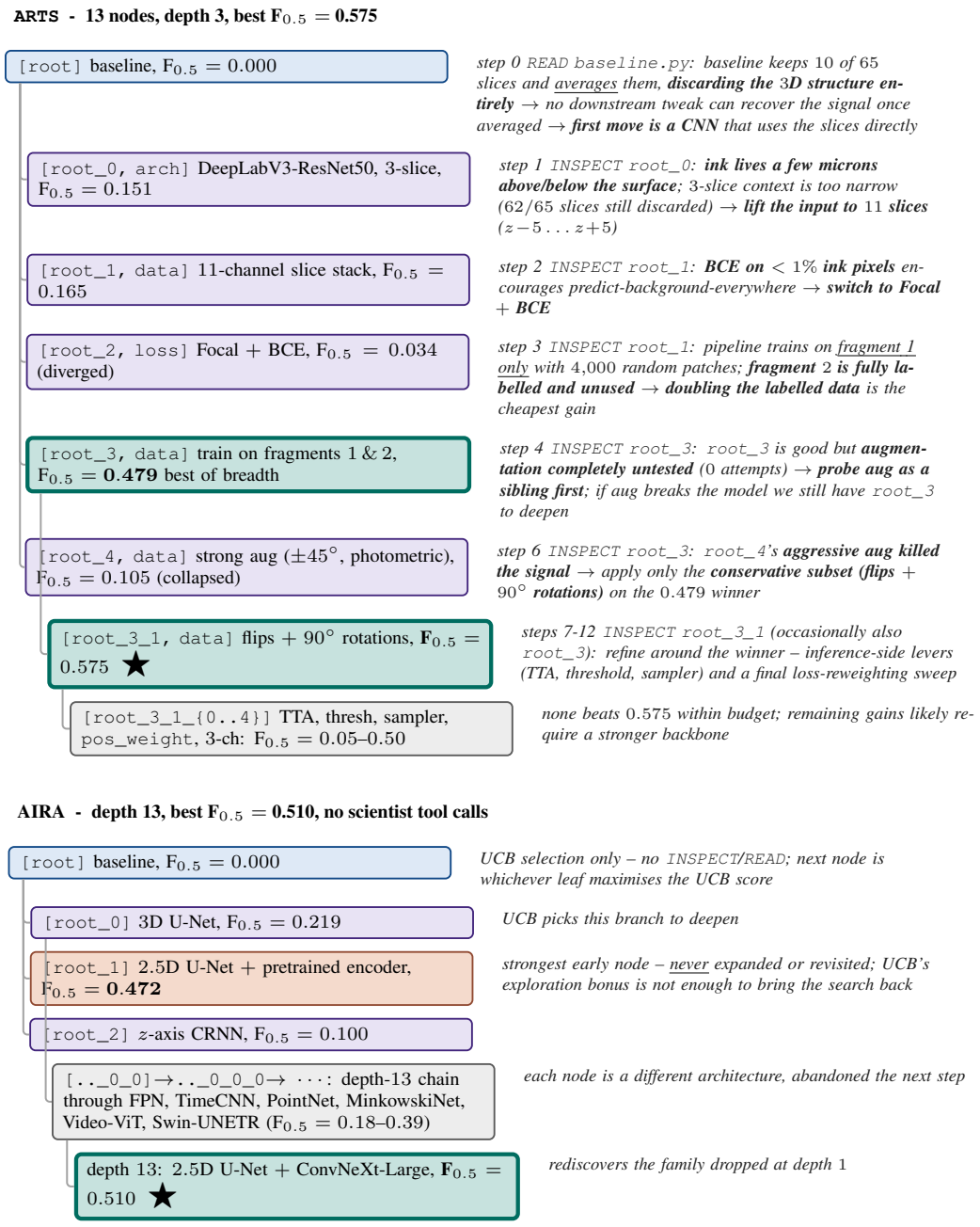


Figure 9: Vesuvius Ink Detection search trees. ARTS (top) keeps five distinct axes as siblings of root, deepens the dominant lever (root_3, both fragments, 0.479) with augmentation to **0.575**. AIRA (bottom) commits to the 3D-U-Net branch via UCB, abandons a stronger 2.5D-pretrained sibling at depth 1, and only rediscovers that family 12 nodes deeper at 0.510.

that drove the next experiment. ARTS reaches best KL **0.467**; AIRA pivots after its first draft (KL 0.513) and the children of its pivot diverge to KL 4.08 and 9.43, never returning to the working family.

G VESUVIUS SEARCH TRAJECTORY: ARTS VERSUS AIRA

We contrast ARTS and AIRA on Vesuvius Challenge Ink Detection (MLE-bench; $F_{0.5}$, higher is better). Tree nodes are colored **blue** (baseline), **purple** (progress), **teal** (new best, ★), **red** (regression

/ abandoned strong node). Depth is shown by indentation, `par=X` labels, and gray parent→child L-branches. Italic margin notes give the scientist's tool call and the reasoning that drove the next experiment.

H METAMAZE SEARCH TRAJECTORY: ARTS VERSUS AIRA

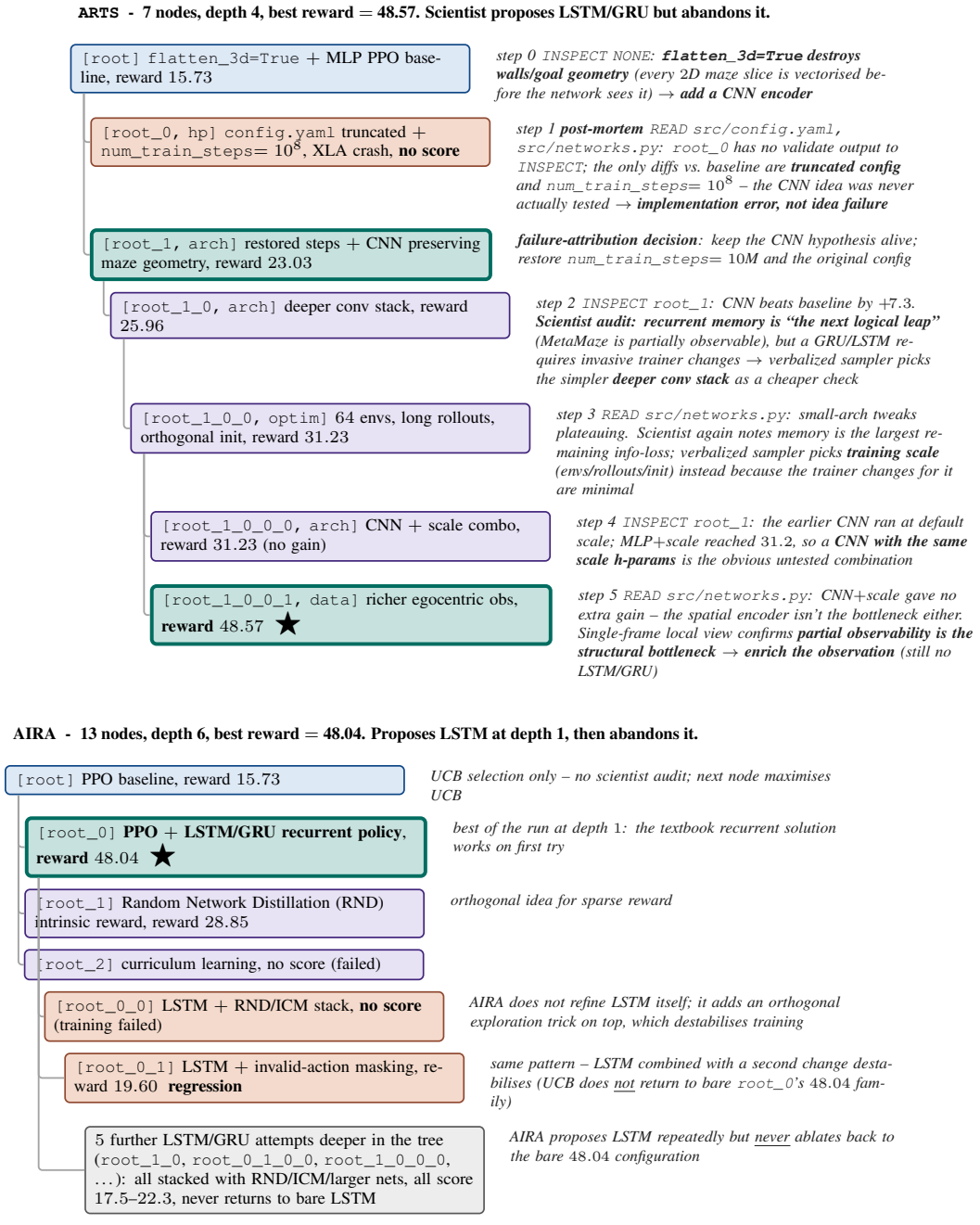


Figure 10: MetaMaze search trees. ARTS (top, 7 nodes) recovers from a config-truncation crash via failure attribution, then climbs 15.73 → 48.57 on a pure-MLP path – the scientist explicitly identifies recurrent memory as the missing axis at steps 2, 3 and 5 but the verbalized sampler always selects a cheaper alternative. AIRA (bottom, 13 nodes) proposes a PPO+LSTM recurrent policy at root_0 and reaches **48.04** on first try, but UCB then deepens elsewhere and never returns to refine the bare LSTM; 5 further LSTM attempts deeper in the tree all combine LSTM with RND/ICM/larger nets and destabilise at 17.5–22.3. Both methods peak near 48 for opposite reasons; the full LSTM gain (48 →≈90) is recovered by test-time training (Table 2, MetaMaze: 30.4 → **53.0**).

We contrast ARTS and AIRA on MLGym MetaMaze (mean evaluation reward, higher is better; ceiling ≈100). Both methods reach nearly identical peaks (48.57 for ARTS, 48.04 for AIRA), but

for opposite reasons. MetaMaze is partially observable, so a recurrent policy (LSTM/GRU) is the textbook solution. ARTS's scientist repeatedly identifies recurrent memory as "the next logical leap" in its reasoning but the verbalized sampler keeps picking simpler alternatives (deeper conv, training scale, richer observation), so no LSTM/GRU node is ever actually run. AIRA proposes LSTM at root_0 and reaches the run's best (48.04) on its first try, then deepens the wrong branch, abandons root_0, and stacks RND/ICM on top of 5 further LSTM attempts which all destabilise at 17.5–22.3.

I MountainCar Search Trajectory: ARTS versus AIRA

ARTS and AIRA on MLGym MountainCarContinuous (mean reward, higher is better; ceiling ≈ 99). Peak scores are similar (~ 95) but ARTS reaches reward 95.73 in 4 steps at depth 4; AIRA reaches 94.82 at depth 13 after 39 expansions ($\sim 10\times$ the compute), with one root draft diverging to $-9{,}708.99$. The MountainCar margin in Table 2 is primarily a robustness gap (mean ARTS $95.73_{\pm .69}$ vs. mean AIRA $80.82_{\pm 11.75}$). Italic margin notes give the scientist's tool call and the reasoning that drove the next experiment.

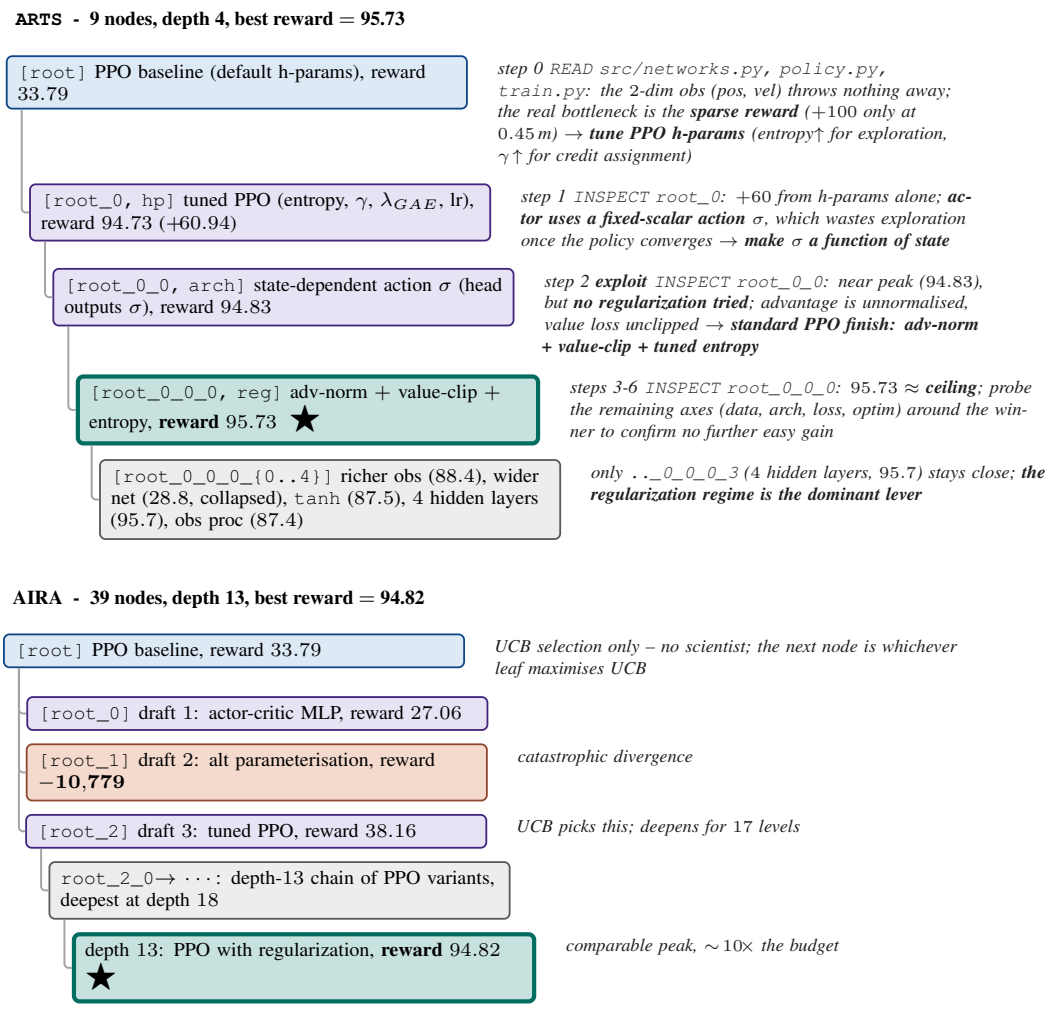


Figure 12: MountainCar search trees. ARTS (top) makes three sequential moves — tune PPO h-params, add state-dependent action σ, add regularization — and reaches reward $\mathbf{95.73}$ at depth 4. AIRA (bottom) drafts three roots; one diverges catastrophically, UCB commits to the 38.16 draft, chains 17 further levels to 94.82. The table margin reflects baseline runs that frequently diverge.

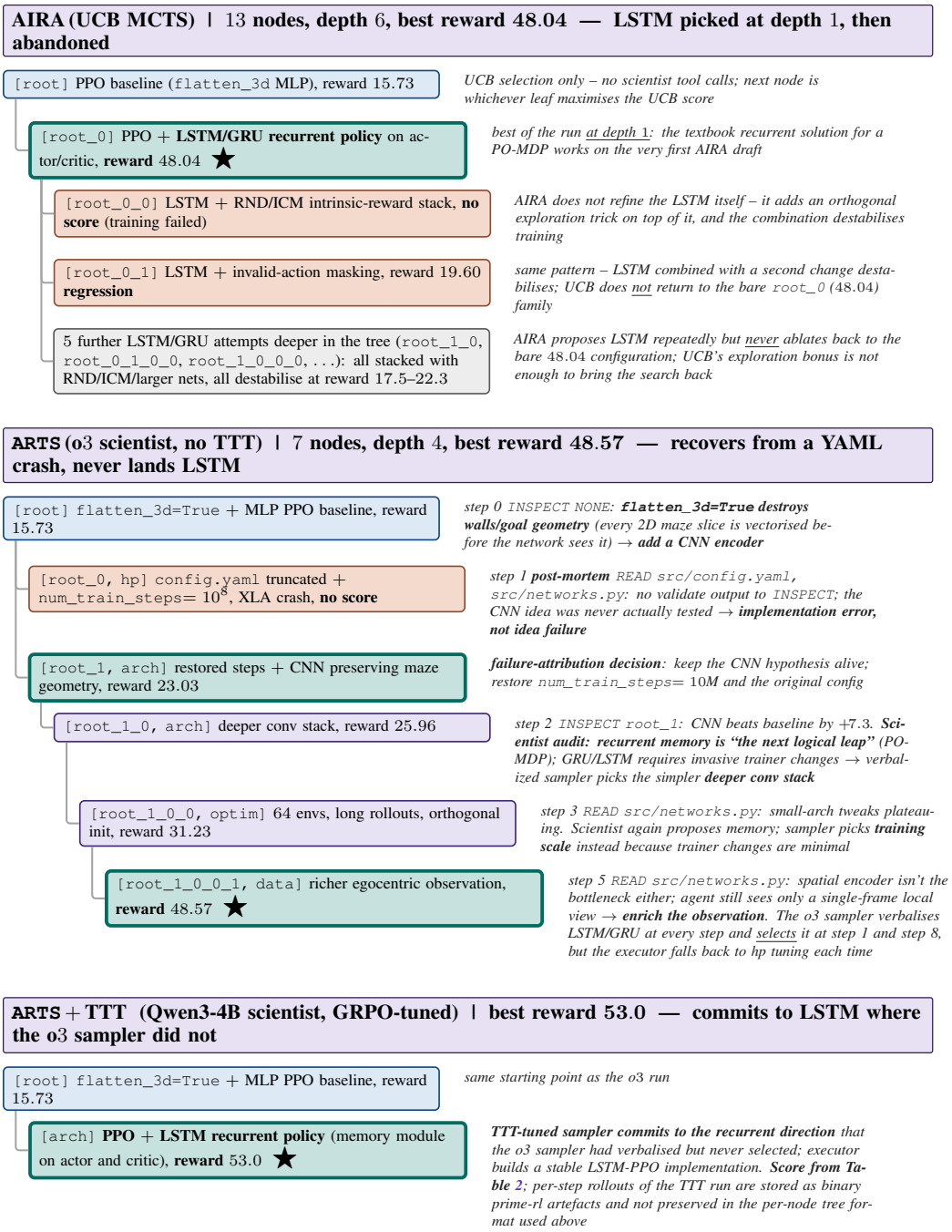


Figure 11: **MetaMaze: AIRA, ARTS-o3, and ARTS+TTT search trees on the same task and budget.** Nodes are coloured blue (baseline), purple (progress), teal (new best, ★), and red (regression / abandoned). AIRA picks LSTM at depth 1 (48.04) and never refines it. ARTS-o3 recovers from a crash via failure attribution and climbs 15.7 → 48.57 on an MLP path; it tries LSTM but abandons it. ARTS+TTT commits to the recurrent direction and reaches **53.0**. Full appendix versions in Appendix H.

J Qualitative Analysis of Search Efficiency

This section studies whether ARTS searches more efficiently than the baselines. We use the logs to ask the following questions.

1. Can ARTS separate a promising hypothesis from a bad execution?
2. Does auditing identify the root cause of failed or regressed experiments?
3. Does the scientist reason from logs rather than only follow final scores?
4. Does reasoning-based parent selection spend budget better than heuristic selection?
5. Can the scientist propose directions that are not already in the tree?
6. Does verbalized sampling preserve enough diversity?
7. How do baselines fail differently under the same budget?

We answer each question separately below.

J.1 Can ARTS separate a promising hypothesis from a bad execution?

Answer: yes. The logs show that ARTS often preserves a promising hypothesis after a flawed implementation. On HMS Brain Activity, the task is to predict seizure-related activity from EEG spectrograms, and a central difficulty is how to aggregate multiple windows for the same patient. Using more spectrogram windows is promising because the single-window baseline discards much of the labeled signal. The first all-window attempt regressed from 0.963 to 1.237 KL. ARTS did not abandon the idea. It kept the same broad hypothesis and changed the implementation. On Vesuvius, where the task is to segment hidden ink from 3D scroll scans, an EMA regularization attempt returned 0.000 $F_{0.5}$ because the model predicted an all-zero mask. ARTS treated this as a broken EMA implementation, not as evidence that EMA was a bad idea. On FineWeb language modeling, a longer-training attempt timed out after reducing the per-device batch size. ARTS kept the longer-training hypothesis and later moved validation loss from 4.67 to 4.29 and then to 3.97.

J.2 Does auditing identify the root cause of failed or regressed experiments?

Answer: yes. The audit is useful when the score alone is ambiguous. In HMS, the failed all-window node looked like a bad direction from the metric alone. The audit instead found two concrete implementation errors: duplicated `eeg_id`s in the split and averaging probabilities after softmax instead of aggregating logits before softmax. In Vesuvius, the all-zero prediction under EMA was traced to how the EMA model was maintained and evaluated. The next proposal maintained shadow weights after each optimizer step and used the EMA model only for validation and inference. In FineWeb, the timeout was diagnosed as a throughput problem, not as evidence against training longer. These diagnoses are specific enough to change the next experiment.

J.3 Does the scientist reason from logs rather than only follow final scores?

Answer: yes. Several decisions depend on logged failure modes rather than the final score alone. On FineWeb, the scientist inspected the training curves and concluded that the model was under-trained, so it extended training instead of only sweeping batch size or attention heads. On APTOS, a naive resolution increase to 384^2 ran successfully but reduced QWK from 0.932 to 0.906. ARTS did not keep increasing resolution. It reasoned that resolution would only help with better cropping, augmentation, or regularization. On HMS, replacing ResNet18 with ResNet34 under the same pipeline worsened KL from 0.541 to 0.627. Since the run was valid, the scientist treated this as evidence against simply scaling the backbone. These are log-level inferences, not score-only choices.

J.4 Does reasoning-based parent selection spend budget better than heuristic selection?

Answer: yes, by both qualitative evidence and automatic proxies. A score alone cannot tell whether a node failed because the idea was weak or because the implementation was bad. ARTS uses inspection, memory, and reasoning to choose which parent deserves another child. This lets it return to a promising branch after an implementation failure, while avoiding branches that ran cleanly but regressed. We also compute a rough proxy for node-selection quality. We define a

valid-but-regressing node as a node that produces a valid score, does not have a fatal execution marker, and is worse than its parent. Such nodes are an imperfect proxy for bad ideas with good execution. On selected tasks with known metric direction, ARTS expanded only 1.4% of valid-but-regressing nodes, compared with 92.7% for AIRA and 64.2% for MLEvolve. The mean number of descendants below such nodes was 0.76 for ARTS, 30.3 for AIRA, and 6.23 for MLEvolve. Among internal nodes, the fraction that eventually led to an improving descendant was 80.4% for ARTS, compared with 75.8% for AIRA and 64.1% for MLEvolve. These numbers are noisy because execution status is not uniformly logged across methods, but they support the qualitative pattern that ARTS spends less budget on cleanly executed but unpromising branches.

J.5 Can the scientist propose directions not already in the tree?

Answer: yes. The scientist is not restricted to mutating the current best code. It can read the tree, inspect logs, consult memory, and propose a new hypothesis before the executor writes code. On HMS, after a strong ResNet18 branch, the scientist did not simply continue local tuning. It listed three directions: combine full-window coverage with the strong backbone, try a larger pretrained architecture, and change the loss for class imbalance. These candidates were assigned probabilities of 0.50, 0.30, and 0.20, and the sampled experiment combined the relevant branches. This is evidence that the scientist can propose directions not already represented by the current best lineage.

J.6 Does verbalized sampling preserve enough diversity?

Answer: yes. Verbalized sampling forces the scientist to list multiple candidate hypotheses with explicit probabilities before the system samples one. The probabilities represent the scientist's current belief about which hypotheses are worth trying after inspecting the tree, logs, and memory. This makes the proposal distribution visible and keeps the search from collapsing to a single local edit. We measure rough strategy coverage with keyword labels over node hypotheses. ARTS visits 4.11 unique axes per run on average, compared with 2.20 for AIRA and 2.43 for MLEvolve. The axis-distribution entropy is 1.16 for ARTS, 0.43 for AIRA, and 0.58 for MLEvolve. These numbers support the claim that verbalized sampling increases diversity. However, diversity alone is not enough. The stronger finding is that ARTS combines diversity with inspection: it opens new axes when the audit or memory says an axis is missing, and it returns to old parents when a promising idea failed for an execution reason.

J.7 How do baselines fail differently under the same budget?

Answer: the baselines fail in method-specific ways. Linear search is brittle because it continues the last node and has no explicit mechanism to return to an older promising parent after a bad implementation. AIRA can sometimes describe that a failure is likely a bug, but the diagnosis is less reliably coupled to the next parent choice. In HMS, AIRA expanded an unstable spectrogram-CNN branch whose score had already degraded to 4.08 KL; the next child worsened further to 9.43 KL. On FineWeb, AIRA produced root-level attempts without a valid submission, including a CUDA out-of-memory failure during validation, and did not recover a scored branch from that failure. MLEvolve explores through mutation and crossover, which can be useful for compact programs, but its reflections are sometimes weakly grounded in the code that actually ran. In MetaMaze, for example, it sometimes attributes performance to recurrent LSTM/GRU memory even when the logged branch still uses a flattened feedforward representation. These failures are not just lower scores. They show weaker coupling between evidence, parent selection, and the next executable experiment.

K Extended Related Work

Autonomous scientific discovery. Recent systems use language models as proposal engines for scientific discovery. FunSearch (Romera-Paredes et al., 2024) combines an LLM with an evaluator in an evolutionary loop and discovers new constructions in extremal combinatorics. AlphaEvolve (Novikov et al., 2025) uses Gemini as a code mutation engine and reports improvements in algorithmic domains such as matrix multiplication. Coscientist (Boiko et al., 2023) and Google's AI Co-Scientist (Gottweis et al., 2025) study hypothesis generation and planning in wet-lab settings. The Sakana AI Scientist (Lu et al., 2024) automates parts of the ML research pipeline, including hypothesis generation, code writing, execution, and write-up. These systems differ from our setting in both domain and search structure. We study ML research tasks where each candidate requires writing and running training code, and where the main question is how to choose which prior node to expand.

Linear agents. MLGym (Nathani et al., 2025) and MLAgentBench (Huang et al., 2024) evaluate agents that interact with ML workspaces through sequential actions. AutoResearch (Karpathy, 2026) adds planning, but still follows a single trajectory. This structure is computationally simple and easy to deploy, but it commits strongly to early decisions. Reasoning models can backtrack within a chain of thought (Guo et al., 2025; Qin et al., 2025), but the revision happens inside the current context window. In long ML runs, the agent accumulates code, logs, validation scores, and failure traces. Prior work on long-context models shows that information retrieval and reasoning degrade as relevant evidence is pushed deeper into context (Liu et al., 2024; Hsieh et al., 2024). This makes linear search fragile when the useful direction is an earlier hypothesis that failed for implementation reasons.

Tree search. Tree-based agents keep an explicit search tree and choose a node to expand. Greedy variants such as AIDE (Jiang et al., 2025) expand the best-scoring node, while MCTS variants such as AIRA (Toledo et al., 2025) and AI Scientist v2 (Yamada et al., 2025) use UCT-style selection. These methods import a useful abstraction from games, but ML research does not have cheap or reliable rollouts. A node score is a noisy mixture of hypothesis quality, implementation correctness, data handling, training stability, and evaluation details. A valid idea can receive a poor score because of a coding error, and a weak idea can receive a high score because the implementation is easier. Score-based backpropagation therefore has the wrong primitive for deciding which scientific direction deserves more attempts.

Evolutionary search. Evolutionary and population-based systems such as FunSearch (Romera-Paredes et al., 2024), AlphaEvolve (Novikov et al., 2025), OpenEvolve (Sharma, 2025), and MLEvolve (Du et al., 2026) maintain a population of programs and apply mutation or crossover. This is effective when the artifact being searched is short and compositional. ML training pipelines are less modular. Data loading, model architecture, losses, augmentations, optimizers, and validation logic interact in tightly coupled ways. Copying a block from one branch into another can break hidden assumptions rather than combine useful ideas. Evolutionary search also spends many evaluations on program variants, which is expensive when each node can require minutes to hours of GPU time.

LLM-guided search. Several works use language models to replace or augment hand-designed search heuristics. Verbalized sampling (Zhang et al., 2025) asks the model to express a distribution over candidates, which can reduce collapse to a single likely continuation. We use this idea when the scientist proposes hypotheses, because greedy continuation often produces small variants of the current best branch. The key difference in `ARTS` is that the scientist also selects the parent node. It can inspect logs and code from prior attempts and distinguish a failed implementation from a failed hypothesis.

Test-time training. Test-time training was introduced as a way to adapt a model on each test instance (Sun et al., 2020). Parameter-efficient variants use adapters such as LoRA to update only a small set of weights (Akyürek et al., 2025). TTT-Discover (Yuksekgonul et al., 2026) and execution-grounded agent training (Si et al., 2026) apply on-policy optimization to research agents, using rollout outcomes as the learning signal. Our setting differs in the structure of the policy. We train a tree-structured scientist that chooses what to inspect, which parent to expand, and which hypothesis to test. This lets the model absorb useful search experience into its weights while still using execution feedback from the current task.