MLS-Bench: A Holistic and Rigorous Assessment of AI Systems on Building Better AI

Bohan Lyu [*1] **Yucheng Yang** [*2] **Siqiao Huang** [*2] **Jiaru Zhang** [*3] **Qixin Xu** [*1] **Xinghan Li** [*4] **David Han** [*1] **Huaqing Zhang** [*2] **Yicheng Zhang** [*2] **Runhan Huang** [5] **Kaicheng Yang** [6] **Zitao Chen** [2] **Wentao Guo** [7] **Junlin Yang** [2] **Xinyue Ai** [8] **Wenhao Chai** [7] **Yadi Cao** [9] **Ziran Yang** [7] **Kun Wang** [7] **Dapeng Jiang** [2] **Huan-ang Gao** [2] **Shange Tang** [7] **Chengshuai Shi** [7] **Simon S. Du** [4] **Max Simchowitz** [10] **Jiantao Jiao** [1] **Dawn Song** [1] **Chi Jin** [7]

Abstract

Modern AI progress has been driven by ML methods that are generalizable across settings and scalable to larger regimes. As large language models demonstrate advanced capabilities in reasoning, coding, and engineering tasks, it is increasingly important to understand whether they can discover such methods rather than only apply existing ones. We introduce MLS-Bench, a benchmark for evaluating whether AI systems can invent generalizable and scalable ML methods. MLS-Bench contains 140 tasks across 12 domains, each requiring an agent to improve one targeted component of an ML system or algorithm and demonstrate that the improvement generalizes across controlled settings and scales. We find that current agents remain far from reliably surpassing human-designed methods, and that engineering-style tuning is easier for them than genuine method invention. We further study the effects of test-time scaling, adaptive compute allocation, and context provision on agents' discovery performance, together with case studies of their behavior. Our analyses suggest that the bottleneck is not only in proposing new methods, but also in the scientific insight needed to plan, validate, and scale claims about them. More search, compute, or context alone does not remove this bottleneck. We build and maintain a community platform for cumulative and comparable iteration, and release the data and code at https://mls-bench.com.

[*]Core Contributor [1]UC Berkeley [2]Tsinghua University [3]Purdue University [4]University of Washington [5]Harvard University [6]Shanghai Jiao Tong University [7]Princeton University [8]University of Pennsylvania [9]UC San Diego [10]Carnegie Mellon University.

ICML 2026 AI4Math Workshop, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

1. Introduction

"We want AI agents that can discover like we can, not which contain what we have discovered."

— Richard S. Sutton, *The Bitter Lesson*

Large language models (LLMs) have evolved from chatbots (Brown et al., 2020; OpenAI, 2023; Bai et al., 2023; Team, 2025a; Touvron et al., 2023; Team, 2023; Chiang et al., 2024) into agents that pursue long-horizon objectives (Schick et al., 2023; Yao et al., 2023; Zhou et al., 2024; Mialon et al., 2024; OpenAI, 2025; Yang et al., 2025a; Guo et al., 2025; Team, 2025b; Anthropic, 2025; Comanici et al., 2025), including software engineering (Jimenez et al., 2024; Mündler et al., 2024; Chen et al., 2026a; Liang et al., 2026), deep research (Shao et al., 2025; Schmidgall et al., 2025; Tang et al., 2025; Wei et al., 2025), and mathematical theorem proving (Hubert et al., 2025; Lin et al., 2025a; Chen et al., 2025b; Lin et al., 2025b; Wang et al., 2025a). Attention has recently shifted to more frontier problems including open optimization tasks such as circle packing (Novikov et al., 2025; Yuksekgonul et al., 2026; Wang et al., 2025b; Mang et al., 2025; Sharma, 2025) and machine learning engineering, where agents compete on Kaggle-style tasks (Chan et al., 2025; Huang et al., 2024; Qiang et al., 2025; Panapitiya et al., 2025; Yang et al., 2025b). However, even these more advanced settings still do not resemble how human computer scientists discover new general methods.

This mismatch comes from the target of evaluation. Most agent benchmarks reward *engineering*: improving one fixed instance through data processing, tuning, debugging, and model selection. ML science asks for a method-level idea, such as a new architecture, objective, component, or optimizer, that can be validated beyond the setting that produced it (Simon, 1988; Vincenti et al., 1990; Krenn et al., 2022; He et al., 2016; Vaswani et al., 2017; Ho et al., 2020; Mnih et al., 2015; Schulman et al., 2017; Radford et al., 2021; Zhang & Sennrich, 2019; Dao et al., 2022; Kingma & Ba, 2014; Jordan et al., 2024; Liu et al., 2025a). The question is

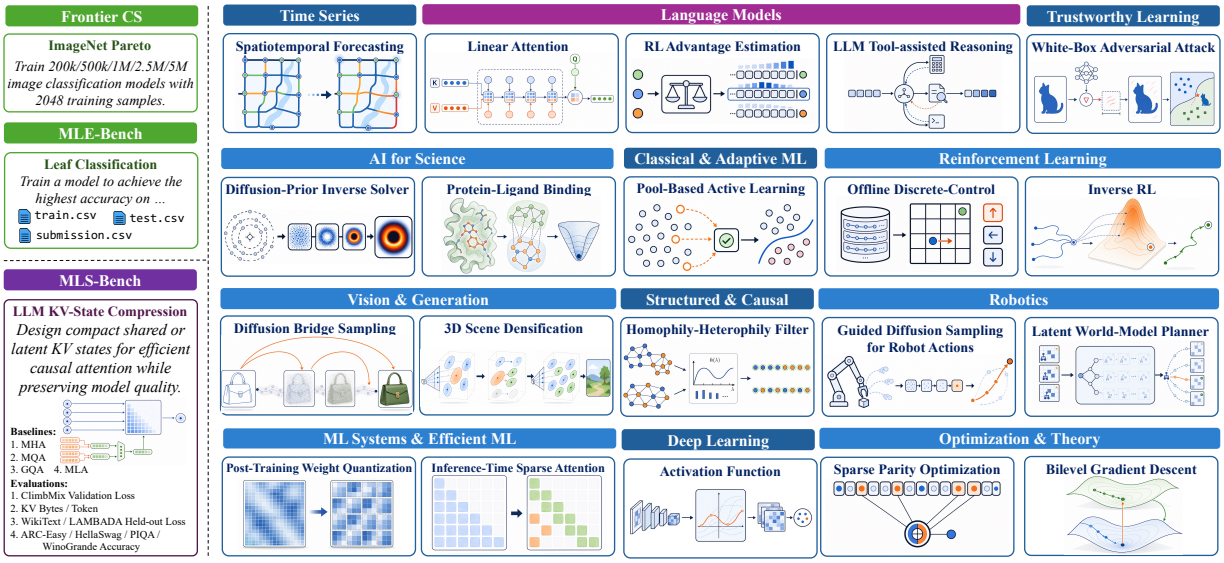

Figure 1. MLS-Bench overview. Left: comparison of Frontier-CS, MLE-Bench, and MLS-Bench tasks. Right: 20 representative MLS-Bench tasks from 140 tasks across 12 domains.

whether agents can create such methods, rather than merely optimize one fixed benchmark or dataset.

Existing benchmarks do not yet isolate this capability. ML-engineering benchmarks mix method choice with implementation and tuning (Chan et al., 2025; Huang et al., 2024; Qiang et al., 2025; Panapitiya et al., 2025; Huaizheng Zhang*, 2024; Yang et al., 2025b); end-to-end research benchmarks make attribution hard (Chen et al., 2025a; Starace et al., 2025); and recent narrow discovery benchmarks remain tied to single components or subfields (Vitvitskyi et al., 2026; Ouyang et al., 2025; Rank et al., 2026; Chen et al., 2026c).

To this end, we introduce **MLS-Bench** for ML Science, a benchmark containing 140 tasks across 12 ML domains that evaluates whether AI systems can produce genuine, transferable ML method improvements. As shown in Figure 1, each task asks an agent to improve a targeted component under controlled edit scopes, reproduced strong human baselines, and multiple evaluation settings. This design makes score gains attributable to the intended method rather than to evaluator changes, training-protocol hacks, or scale increases. We also curate **MLS-Bench-Lite**, a 30-task challenging subset covering all 12 areas for rapid iteration and broader model tracking.

Our evaluation shows that current frontier agents remain far from reliably matching human-designed methods inside the same scaffold, even with strong baselines in context and multiple opportunities to iterate. They are noticeably better at engineering-style tuning than at producing a new method that survives controlled validation, which makes MLS-Bench a demanding target for future foundation models and self-evolving frameworks.

We further study the influence of stronger inference-time support, more context, or greater freedom to experiment. These analyses show that current agents' limitation goes beyond proposing methods: they can search, tune, and recombine familiar ingredients, but struggle with the scientific judgment needed to form hypotheses, design experiments, allocate limited trials, and turn feedback into evidence for scalable claims. Human expert assessment also finds that genuinely new proposals are rare and often weakly justified.

We maintain MLS-Bench as a community benchmark with a growing leaderboard to guide the development of future foundation models and agent harnesses.

2. Related Work

Automated scientific discovery. Computational methods have contributed to scientific discovery across diverse domains (Udrescu & Tegmark, 2020; Lam et al., 2023; Appel & Haken, 1977; Hubert et al., 2025; Romera-Paredes et al., 2024; Jumper et al., 2021; Watson et al., 2023; Swanson et al., 2024; Coley et al., 2019; Merchant et al., 2023), including computer science itself, spanning algorithms and systems (Mirhoseini et al., 2021; Chen et al., 2018; Mankowitz et al., 2023), and especially machine learning—including model architectures (Zoph & Le, 2017; Liu et al., 2019), training procedures (Andrychowicz et al., 2016; Finn et al., 2017; Hutter et al., 2019), and data and loss design (Cubuk et al., 2019; Gonzalez & Miikkulainen, 2020). More recently, LLMs have accelerated automated discovery across a broad spectrum: serving as collaborative scientific partners (Gottweis et al., 2025; Schmidgall et al.,

Table 1. Comparison with representative benchmarks. ✓ means explicit support, △ means partial or indirect support, and ✗ means absent. **Count**: source-reported primary evaluated units. **#**: number of coverage categories. **Scope**: how coverage is categorized, e.g., by repos, domains, tasks, or tracks. **New Method**: creation of a new scientific method rather than replication. **Generalize**: evaluation of whether the same method or artifact works across multiple settings. **Scalability**: scale-sensitive tasks or scalable evaluation design. **Control**: editable or submitted artifact restricted to the problem-relevant part under frozen evaluation.

Benchmark	Setting	Count	#	Scope	New Method	Generalize	Scalability	Control	Reference
ML-Bench (Tang et al., 2023)	ML code	169	18	repos	✗	✗	△	△	Pass@K
MLAgentBench (Huang et al., 2024)	ML experimentation	13	4	categories	✗	✗	✗	✗	baselines
MLE-bench (Chan et al., 2025)	ML engineering	75	15	categories	✗	✗	✓	✓	medals
MLE-Dojo (Qiang et al., 2025)	ML engineering	200+	4	domains	✗	✗	✓	△	H-Rank
PostTrainBench (Rank et al., 2026)	LLM post-training	28	7	evals	✗	△	✓	✓	instruct
AutoResearch (Karpathy, 2025)	LLM training	1	1	setup	△	✗	△	△	val_bpb
SimpleTES (Ye et al., 2026)	Scientific discovery	21	6	domains	✓	△	✓	△	SOTA
MLGym (Nathani et al., 2025)	ML experiments	13	4	domains	△	✗	△	△	baselines
PaperBench (Starace et al., 2025)	Paper replication	20	1	AI	✗	✗	✓	△	rubric
MLR-Bench (Chen et al., 2025a)	Research workflow	201	9	topics	✓	✗	✓	✗	judge
DiscoveryBench (Majumder et al., 2025)	Data discovery	1167	6	domains	✗	✗	✓	✓	facets
ScienceAgentBench (Chen et al., 2025c)	Data workflow	102	4	fields	✗	✗	△	✓	papers
AstaBench (Bragg et al., 2025)	Research assistance	2400+	4	areas	△	✗	△	△	baselines
AutoLab (AutoLab Team, 2026)	Research loops	23	3	categories	△	✗	△	✓	medals
AIRS-Bench (Lupidi et al., 2026)	ML SOTA tasks	20	7	categories	✗	✗	✓	✓	SOTA
FIRE-Bench (Wang et al., 2026)	Claim rediscovery	30	1	ML	✗	△	△	△	claim-F1
KernelBench (Ouyang et al., 2025)	Kernel optimization	250	3	levels	✗	✗	△	✓	PyTorch
ALE-Bench (Imajuku et al., 2025)	Algorithm optimization	40	10	genres	△	✗	✓	✓	leaderboard
FrontierCS (Mang et al., 2025)	CS problems	156	2	tracks	✓	△	△	✓	expert
MLS-Bench	**ML science**	**140**	**12**	**domains**	✓	✓	✓	✓	baselines

2025), optimizing specific algorithms and computational components (Novikov et al., 2025; Romera-Paredes et al., 2024; Ouyang et al., 2025), and driving fully autonomous research (Lu et al., 2024; Yamada et al., 2025; Tang et al., 2025). A growing set of benchmarks evaluates these emerging capabilities (Majumder et al., 2025; Chen et al., 2025c; Liu et al., 2025b; Panigrahi et al., 2026; Bragg et al., 2025).

Self-evolving agents and evaluation. The paradigm of LLMs has evolved from single-turn question answering (Brown et al., 2020) toward agents that iterate over extended horizons (Yao et al., 2023; Schick et al., 2023). Self-evolving systems iteratively refine solutions through evolutionary search (Novikov et al., 2025; Romera-Paredes et al., 2024; Chen et al., 2026b; Si et al., 2026), open-ended self-improving loops (Lange et al., 2025; Assumpção et al., 2025; Panfilov et al., 2026; Karpathy, 2025), and test-time training (Yuksekgonul et al., 2026; Phan et al., 2025; Wang et al., 2025b; Surina et al., 2025; Zuo et al., 2025). However, these systems have been demonstrated primarily on specific optimization problems, such as circle packing, contest-style algorithm search, kernel optimization, and activation-function search (Novikov et al., 2025; Yuksekgonul et al., 2026; Wang et al., 2025b; Vitvitskyi et al., 2026). Such settings are narrow in domain and do not capture whether a discovery is scalable and generalizable.

Benchmarking LLM agents for Coding and ML. Evaluation of LLMs on coding has progressed from code generation (Chen et al., 2021; Jain et al., 2025; Jimenez et al., 2024; Mündler et al., 2024; Tang et al., 2023; Chen et al., 2026a) toward code as a means to broader goals, including ML engineering (Huang et al., 2024; Chan et al., 2025; Qiang et al., 2025; Panapitiya et al., 2025) and open-ended scientific research (Wijk et al., 2025; Nathani et al., 2025; Garikaparthi et al., 2026; Wang et al., 2025c; Lupidi et al., 2026; Wang et al., 2026; Mang et al., 2025). While ML engineering evaluation is well-established, attempts to evaluate ML *science*, i.e., whether AI can produce genuine method-level innovations, face limitations. End-to-end research benchmarks (Chen et al., 2025a) evaluate holistic workflows from ideation to manuscript, but their success criteria are broad, making it difficult to attribute individual method contribution. Other benchmarks target specific ML components (Rank et al., 2026; Anthropic, 2026; Chen et al., 2026c; Ouyang et al., 2025; Press et al., 2025; Imajuku et al., 2025; Vitvitskyi et al., 2026), leaving cross-domain generalization unmeasured.

MLS-Bench instead evaluates generalizable and scalable ML invention. Table 1 compares MLS-Bench with 19 representative benchmark datasets along these dimensions.

3. MLS-Bench

MLS-Bench evaluates whether AI systems can produce genuine, transferable algorithmic innovations. The benchmark is guided by the following principles: (1) **Holistic**: the benchmark covers the major areas the ML community actively pursues and their core research tasks. (2) **Atomic**: each task targets a single research question recognized by its research community as a coherent method-level contribution. (3) **Challenging**: every task includes strong human baselines recognized by the relevant community, including

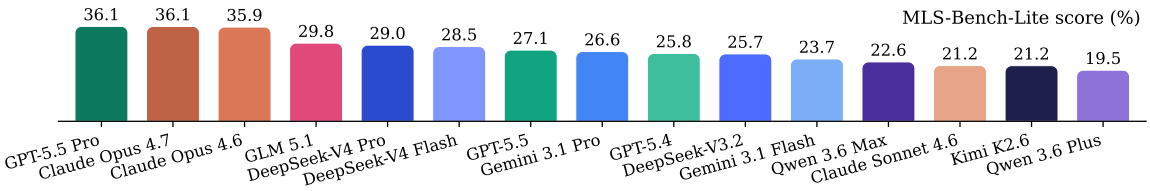

Figure 2. MLS-Bench-Lite Performance across 15 models.

Table 2. Task distribution and representative topics across 12 domains.

Area	Tasks	Topics
Language Models	18	Pretraining, Reasoning RL, Agents, Diffusion LMs
Classical & Adaptive Learning	14	Few-Shot/Meta, Active/Continual/Federated, Calibration
Reinforcement Learning	13	Offline/Online RL, Meta/Multi-Agent RL, Inverse/Safe RL
Optimization & Theory	13	Optimizers, Search/NAS/HPO, Bandits/Bounds
Robotics	12	World Models, Diffusion Policies, Imitation, Control
Vision & Generation	11	3D Vision, Diffusion, VAE/Flow, Image Generation
Deep Learning	11	Architectures, Losses, Augmentation, Normalization
ML Systems & Efficient ML	10	KV Cache, Quantization, Kernels, Sparse Attention
AI for Science	10	Protein, Molecules, Climate/Weather, Inverse Problems
Time Series & Forecasting	10	Forecasting, Imputation/Anomaly, Traffic, Quant Finance
Structured & Causal Reasoning	10	Causal Discovery, Treatment Effects, Graph Learning
Trustworthy Learning	8	Attacks, Robust Training, Privacy, Unlearning

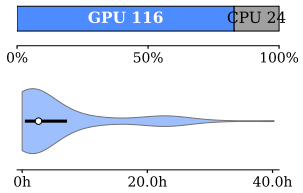

Figure 3. GPU vs. CPU task ratio and the distribution of GPU-hours per experiment.

SOTA methods that we can reproduce. (4) **Generalizable**: solutions are evaluated across multiple settings. (5) **Reproducible**: all runs execute in controlled runtimes with pinned dependencies, fixed seeds, and locked package versions (Section 3.1). (6) **Scientific innovation**: we enforce that performance gains come from the targeted method rather than from modifying the harness or shared training protocols, increasing model capacity, etc. (7) **Scalable**: evaluation scales are chosen to test whether methods remain effective when scaled up (Section 3.2). (8) **Unified scoring**: all metrics are normalized to a bounded scale based on baseline performance, enabling cross-task comparison (Section 3.3).

3.1. Overview

Task Scope. MLS-Bench covers 140 tasks across 12 research areas; Table 2 lists the number of tasks and representative topics in each area. The tasks are built around community-recognized ML-science questions and turn them into executable, controlled, and comparable evaluations. Figure 3 shows the GPU/CPU task split and the distribution of H100 GPU-hours per experiment.

MLS-Bench-Lite. For convenient iteration and broader model tracking, we also curate **MLS-Bench-Lite**, a 30-task subset covering all 12 domains. It keeps the central community-recognized questions in each area while remaining challenging. MLS-Bench-Lite's full list is given in Appendix B. Running all MLS-Bench tasks requires 704.7 H100-hours, while MLS-Bench-Lite requires only 99.2 H100-hours, roughly one day on four H100 GPUs.

Task structure. A task specifies a research problem in executable form. It is defined by (i) a *research question* that describes the research problem, its background and target; (ii) underlying *codebase* with designated *editable scopes* that constrain the regions the agents are able to edit; (iii) at least 3 strong human *baselines* including the SOTA ones that we can reproduce; (iv) at least 3 *evaluation settings* that probe generalization across benchmarks, environments, or base-model scales; (v) a *seeds policy* that requires multi-seed evaluation for tasks whose scores carry non-negligible variance; (vi) a *score normalization* that aggregates all metrics across all settings into a single comparable task-level score; and (vii) a *capacity budget* that caps the agent's model size relative to the baseline when the task includes the modification of model components. The detailed contents of each task are listed in Appendix A.

Infrastructure. To ensure stable reproduction across diverse environments, the evaluation framework is built on a unified backend that supports multiple runtimes, including Apptainer, Docker, Conda, and Harbor (Harbor Framework Team, 2026). At the start of each run, the agent receives the task description, action and test budgets, task-relevant codebase files, and complete baseline implementations. Agents interact through four tools: `edit` modifies allowed code, `test` runs our harness and returns training and visible-test metrics, `submit` selects a previous test result as final, and `undo` reverts edits. See Appendix C for the full system prompt, initial-prompt template, and tool schemas.

3.2. Evaluation Rigor

We employ several strategies to ensure that MLS-Bench reflect genuine method invention rather than confounders: 1. we constrain the agent's search to the algorithmic component under study while keeping the editable scope expressive enough to admit legitimate new methods (Section 3.2.1); 2. we select evaluation scales that preserve scalability evidence

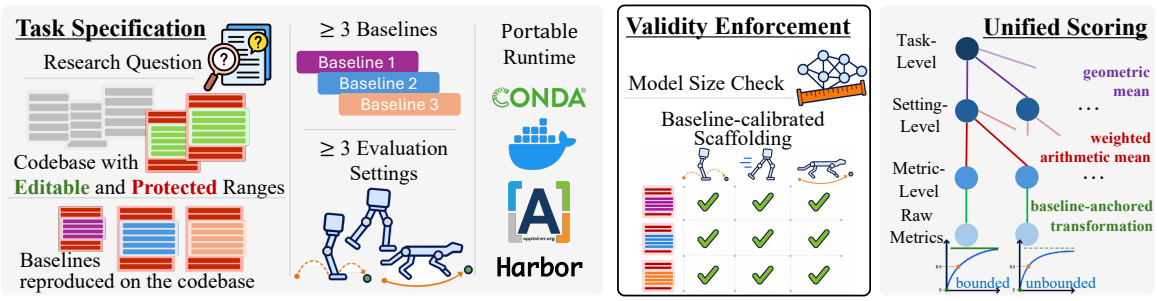

Figure 4. MLS-Bench's design: task specification, validity enforcement, and unified scoring.

under a feasible compute budget (Section 3.2.2); and 3. we guard against contamination and plagiarism (Section 3.2.3).

3.2.1. Isolating the algorithmic axis

An agent can raise its score by inventing a better method, but also by rewriting the evaluation harness, exploiting hyperparameters shared across methods, or inflating model capacity. MLS-Bench mechanically closes the latter ones so that only method invention is rewarded.

Scoping the method. The editable scope of each task is restricted to the component under study, e.g., an architecture block or a training objective, while the evaluation harness remain frozen. Within this scope, we further differentiate between two kinds of hyperparameters: training-protocol knobs shared across methods (epochs, batch size) are locked into protected ranges so that the agent and every baseline run under the same setup, while method-defining hyperparameters (e.g., learning-rate schedule for an optimizer task) remain editable as part of the method itself. Based on these design choices, any score gain is therefore attributable to the component the task requires to study.

Baseline-calibrated scaffolding. While a loose scope may allow the agent to hack, a tight one may prevent the agent from expressing legitimate new methods. We resolve this with a criterion, *baseline-calibrated scaffolding*, that has two interacting parts: 1. a scope-design rule: the editable scope of each task is set to be exactly wide enough to implement every established strong method for the problem as an edit sequence, and no wider; 2. a validity check: every baseline re-implemented inside this scope must reproduce its published reference performance, otherwise the task setup is rejected and revised. The two parts interact, where the scope rule proposes a candidate setup, the reproduction check certifies or refutes that the scaffold and harness faithfully realize the original problem, and a task enters MLS-Bench only when both hold. This mechanism also removes potential bias caused by framework mismatch in MLS-Bench's evaluation.

Capping model capacity. For tasks whose editable scope includes model components, a parameter-budget check instantiates the agent's model alongside each baseline and rejects submissions exceeding the capacity ceiling, forcing gains to come from method rather than from scale hacking.

Design Choice. *We pin evaluation to the algorithmic axis. Baseline-calibrated scaffolding ensures that the editable scope is tight enough to close hacking routes, while reproducing every strong baseline in the same codebase certifies that this scope remains correct and expressive enough for legitimate methods.*

3.2.2. Scalability and feasibility

The scalability of a method is one of its most crucial features, and it's common that some methods help at small scale but fail to help at large scale (Wen et al., 2025; Yang et al., 2021; Everett et al., 2024; Kaddour et al., 2023; Li et al., 2025). However, computational feasibility is equally critical for benchmark design as excessive evaluation cost limits adoption and reduces the benchmark's utility as an iteration signal for method development. Below we outline how MLS-Bench reasons about and navigates this tension.

Relativity of scale. Scale is inherently relative: for any evaluation scale, a larger one can lie beyond it, and the scaling behavior characterized at one regime can be revised when experiments push to larger ones. Qualitative phenomena such as emergence have been reported past previously-studied scales (Wei et al., 2022), though even their characterization is actively revised (Schaeffer et al., 2023); and compute-optimal laws themselves have been re-derived (Kaplan et al., 2020; Hoffmann et al., 2022) while single power-law fits break in new regimes (Caballero et al., 2023). The design problem is therefore not which exact scale to evaluate at, but how to preserve the strongest evidence for scalability compatible with a feasible compute budget.

Principled scale selection. Our principle is that any setting must reproduce the published ranking of the existing baselines. This keeps the reduced task aligned with the original method-level comparison, so gains over baselines

Table 3. Main results on MLS-Bench, comparing frontier agents with *Human SOTA* across 12 areas.

	LM	Rob	V&G	RL	Sys	Sci	Opt	CAL	DL	TS	SCR	TL	Avg
Human SOTA	43.7	41.7	41.3	35.2	45.6	43.3	50.3	41.3	41.3	42.8	47.6	45.6	43.3
Vanilla													
Claude Opus 4.6	26.9	38.4	12.7	21.8	33.9	26.2	31.3	30.1	36.6	26.5	26.6	26.7	28.1
GPT-5.4	24.1	11.7	7.4	10.6	20.3	13.8	32.4	25.1	29.6	10.9	15.8	31.8	19.5
Gemini 3.1 Pro	39.0	23.2	21.6	10.9	29.7	33.4	42.7	18.6	40.4	16.2	12.4	30.1	26.5
DeepSeek-V3.2	35.3	11.0	5.8	11.0	28.2	4.2	25.1	11.4	31.5	16.1	21.6	16.4	18.1
Qwen 3.6 Plus	22.0	15.3	14.3	14.2	13.6	12.0	23.9	20.6	29.8	5.8	1.9	24.4	16.5
Agent													
Claude Opus 4.6	35.0	45.6	31.2	28.2	41.9	28.2	50.1	36.6	38.5	29.8	39.9	27.6	36.0
GPT-5.4	34.7	21.1	21.4	12.4	22.5	23.1	45.4	34.6	31.6	19.5	28.4	39.4	27.8
Gemini 3.1 Pro	43.8	31.9	34.4	26.9	38.0	34.4	42.9	27.1	44.8	34.7	25.8	33.5	34.9
DeepSeek-V3.2	41.1	30.5	18.2	12.4	28.3	20.7	27.7	18.9	33.7	25.7	32.5	26.2	26.3
Qwen 3.6 Plus	38.1	19.9	22.3	13.1	18.1	17.6	29.9	24.8	30.9	16.8	16.2	31.2	23.2

remain evidence of scalability rather than artifacts of an arbitrary small proxy. We keep native scales when feasible; otherwise, we reduce scale as little as possible to make evaluation tractable, while requiring the reduced setting to pass this ordering check.

Design Choice. *By calibrating compute to preserve baseline ordering, MLS-Bench makes feasible evaluation settings informative about method-level scalability.*

3.2.3. Contamination controls

To prevent agents from succeeding by recalling public solutions rather than by inventing new ones, MLS-Bench adopts two complementary safeguards. (1) Each task contains the strongest established method that we could reproduce as a baseline, therefore a solution that merely retrieves a known method is unlikely to beat it. (2) Web search is disabled during our main experiments.

3.3. Evaluation Metrics

Every task in MLS-Bench is evaluated across multiple *settings*, and each setting reports one or more raw *metrics*. We aggregate metric scores within each setting and then aggregate across settings, producing a single bounded task score that is comparable across tasks.

Per-metric normalization. For each metric, we apply a *baseline-anchored transformation*: the worst baseline anchors 0 and the best baseline anchors 0.5 on the internal $[0, 1]$ scale. Because raw metrics differ in direction and units, we write the oriented metric score as $s(x) = \text{sign} \cdot \text{transform}(x)$, where transform uses one of two baseline calibrations:

$$\text{transform}(x) = \begin{cases} \left(\frac{x - x_{\text{floor}}}{x_{\text{bound}} - x_{\text{floor}}}\right)^{\gamma}, & \text{if } x_{\text{bound}} \text{ exists,} \\ 2\sigma\left(\frac{x - x_{\text{floor}}}{\lambda}\right) - 1, & \text{else.} \end{cases} \quad (1)$$

Here x_{floor} and x_{ref} are the worst and best baselines after applying any metric-specific preprocessing. The parameters $\gamma = \log 0.5 \,/\, \log\big((x_{\text{ref}} - x_{\text{floor}})/(x_{\text{bound}} - x_{\text{floor}})\big)$ and $\lambda = (x_{\text{ref}} - x_{\text{floor}})/\log 3$ are chosen so that $s(x_{\text{ref}}) = 0.5$.

Aggregation across levels. Within a setting, the score is the weighted arithmetic mean of its metric scores, $S_{\text{setting}} = \sum_i w_i s_i / \sum_i w_i$, where w_i is a human-labeled weight. Across settings, we take the geometric mean, $S_{\text{task}} = \left(\prod_k S_{\text{setting},k}\right)^{1/K}$, so a method cannot compensate for failure on one generalization setting by hacking another.

4. Experiments

4.1. Setup

Models. We evaluate five frontier models on the full dataset: `Claude Opus 4.6`, `GPT-5.4`, `Gemini 3.1 Pro`, `DeepSeek-V3.2`, and `Qwen-3.6 Plus`. On MLS-Bench-Lite, we test ten more models: `Claude Opus 4.7`, `Claude Sonnet 4.6`, `GPT-5.5 Pro`, `GPT-5.5`, `Gemini 3.1 Flash Lite`, `DeepSeek-V4 Pro`, `DeepSeek-V4 Flash`, `Qwen-3.6 Max`, `Kimi K2.6`, and `GLM 5.1`. All models use high reasoning effort, a 10,000-token thinking budget, provider-default sampling temperature, fixed seeds, and no web search.

Settings. In the main experiments, each agent is allowed at most 20 actions, including at most 3 `test` calls, before submitting a final proposal. We report two scores: *Vanilla*, the first `test` result, and *Agent*, the final submitted result. For the ten additional MLS-Bench-Lite models, each agent is allowed 8 actions and 1 `test` call, so all MLS-Bench-Lite comparisons use the first-proposal *Vanilla* result. We report arithmetic means across task scores. We report the best human baseline as *Human SOTA*, scored with the same normalization as the agents. **Note that** *Human SOTA* is not fixed at 50 after aggregation: it can fall below 50 when different baselines win different metrics or settings, and can exceed 50 when a baseline reaches a metric's theoretical upper bound, which is mapped to 100. For high-variance tasks, all reported scores are multi-seed means, which give

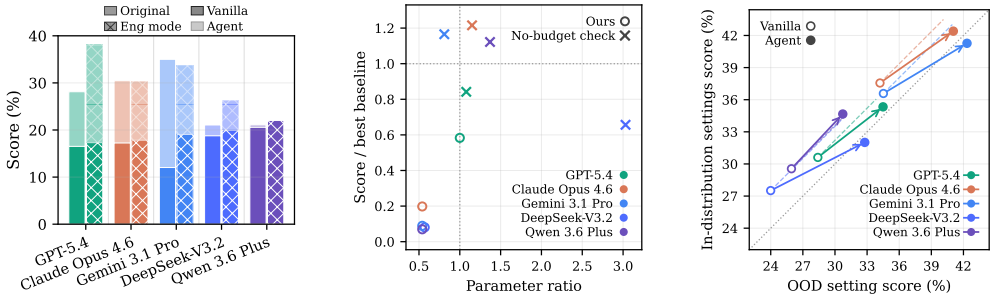

Figure 5. Ablations. **Left:** scientific-innovation vs. engineering-optimization prompt. **Middle:** the budget check prohibits the models from hacking model size for higher performance. **Right:** in-distribution vs. OOD settings, from first proposal to final submission.

stable baseline orderings. Experiments are executed and reproducible on H100 GPUs. Some ablation and analysis experiments are evaluated on a subset where the property under study is well defined, and tasks within each subset are listed in Appendix E.

4.2. Main Results

Table 3 reports per-area scores for the five models under *Vanilla* and *Agent* alongside *Human SOTA*. Even with the full baseline implementations in context, frontier agents usually fail to match the strongest reproduced human methods when asked to implement a new algorithm. Iteration improves many submissions, but it mainly narrows the gap; it does not make current agents reliably competitive with methods already expressible inside the same scaffold. This unsaturated difficulty shows that MLS-Bench offers a durable target for the community to measure future progress.

> ***Takeaway.*** *Even when strong human baseline implementations are provided in context, current agents generally do not reach baseline-level performance. This shows both that MLS-Bench is a challenging target and that models are still weak at building on strong baselines to discover better methods.*

4.3. Ablations

We study three questions behind the evaluation protocol: (1) whether agents are better at inventing new methods or tuning existing methods; (2) the effects of validity controls in our design; and (3) whether iterative refinement transfers across settings.

Scientific innovation versus engineering optimization. Figure 5 (left) compares the scientific-innovation prompt with an engineering-optimization prompt. While `Claude Opus 4.6` and `Gemini 3.1 Pro` remain stable, other models gain especially after several iterations. The contrast shows that those agents, especially weaker models, are stronger at tuning parameters, applying known techniques, and polishing an existing implementation than at proposing a new scientific method.

Validity controls. We evaluate the capacity-budget control on computer vision and reinforcement learning tasks, where agents can adjust the model size. While removing the budget constraint does not consistently improve overall average performance, it opens the door to a recurring shortcut. As illustrated by the specific cases in Figure 5 (middle), models often exploit this lack of restriction by artificially inflating model capacity to trivially surpass human SOTA. Our budget check effectively precludes this hacking behavior. Additionally, we ablate the editable scope. Providing agents with a broader edit space does not enhance their effectiveness; rather, they frequently misuse this flexibility for off-target code modifications, which introduces implementation noise and degrades performance.

Domain generalization. For each task, we annotate one predefined out-of-distribution setting. We then track scores on the OOD setting versus the rest, from first proposal to final submission. Figure 5 (right) shows that, for most models, especially the strong ones, the initial in-distribution-vs-OOD gap shrinks by the final submission. This indicates that iterative refinement genuinely transfers across distributions, and MLS-Bench measures methods that travel rather than ones that merely fit the fixed settings.

> ***Takeaway.***
> *(a) Weaker models are better at engineering optimization than genuine method discovery.*
> *(b) Models tend to hack model parameters for higher scores, making the compatibility check necessary.*
> *(c) Strong models improve across settings while refining iteratively, so our multi-setting evaluation can probe whether a method truly generalizes.*

5. Analysis

Beyond the evaluation above, we study (1) test-time scaling, asking whether more tokens and compute budget can keep producing gains (Section 5.1); (2) adaptive compute allocation, placing agents in a realistic ML-science setting (Sec-

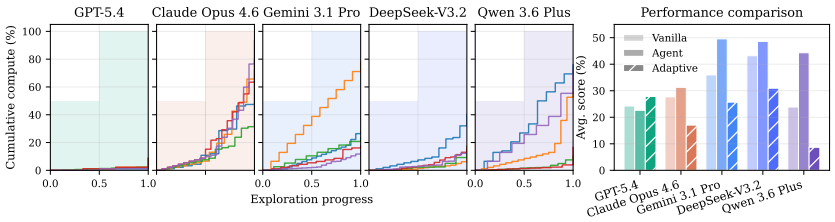

Figure 6. Adaptive compute-allocation experiment. **Left:** cumulative compute budget consumed along the exploration trajectory. **Right:** final-submission scores.

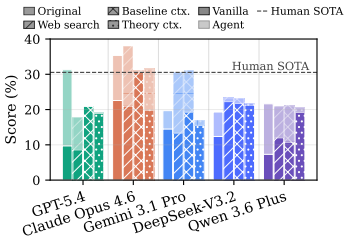

Figure 7. Context engineering.

tion 5.2); (3) context engineering, measuring how additional context changes model behavior (Section 5.3); and (4) human assessment, case studies, and error analysis, diagnosing where agents fail and how they can improve (Section 5.4).

5.1. Test-Time Scaling

Setup. We evaluate four test-time scaling setups on low-latency MLS-Bench tasks: (1) **Scaling Sampling**, which samples many independent first proposals; (2) **Scaling Exploration**, which gives the agent more rounds of iterative experimentation; (3) **Test-time Evolution**, which runs an OpenEvolve population search with islands, mutation, selection, and execution feedback (Sharma, 2025); and (4) **Test-time Training**, which updates the model from experimental feedback based on the TTT-Discover framework (Yuksekgonul et al., 2026). The first three inference-only setups use `Gemini 3.1 Pro`, while the test-time-training setup uses `Qwen3.5-35B-A3B`. For the latter two setups, we make only two of the three settings visible to the model. Details for those setups are in Appendix D.

Results. Scaling helps on simpler tasks: scaling sampling or exploration often raise the best score, but the gains quickly saturate and additional compute stops helping. On the complex deep-learning task, more input does not break the ceiling, let alone exceed Human SOTA. OpenEvolve and test-time training show severe hacking behavior: while their visible-setting scores are maintained or improved, hidden-setting performance declines. These results suggest that test-time optimization needs sufficient setting diversity; otherwise agents improve observed cases rather than transfer.

Takeaway.
(a) Extra test-time compute can improve easier cases, but current scaling methods quickly hit a ceiling; on harder tasks, that ceiling remains below the strongest provided human baseline.
(b) Under partial feedback, scaling can optimize visible settings rather than the underlying method, raising observed scores while damaging hidden-setting performance.

5.2. Verifier-Limited Compute Allocation

The discovery systems above assume fast-verifier regimes. However, ML method discovery is often verifier-limited: proxy runs cannot by themselves establish scalable conclusions, and decisive evaluations are costly. ML scientists work under limited compute budgets. We therefore simulate the setting faced by ML scientists, especially LLM pretraining researchers, where an agent adaptively allocates limited compute across experiments and scales.

Setup. In the main experiment, for pretraining tasks, the standard *Agent* protocol allows three full 345M-parameter runs. In this experiment, we convert the compute of the first two runs into an adaptive budget. Compute is measured as $N \cdot D$, where N is model size and D is training tokens. During exploration, the agent chooses a proxy model size from $\{51\text{M}, 124\text{M}, 199\text{M}, 345\text{M}\}$ and sets the token count for each `test` call. We allow at most 50 actions and 20 `test` calls. We run this experiment on five LLM pretraining tasks.

Results. The adaptive protocol gives agents strictly more experimental choices than *Vanilla* or the fixed *Agent* setting, yet performance generally drops as shown in Figure 6. First, improvement is not monotonic in compute spent: `GPT-5.4` uses little budget yet is the only model that improves, while `Claude Opus 4.6` spends aggressively and still loses. `Gemini 3.1 Pro`, `DeepSeek-V3.2`, and `Qwen-3.6 Plus` follow roughly linear spending trajectories, whereas `Claude Opus 4.6` shows an accelerating, almost exponential pattern.

This result points beyond the method-proposal gap shown above. When agents are given more autonomy to act as ML scientists, they often perform worse. The failure suggests that current models are not only weak at proposing new methods; they also lack the scientific judgment needed to build and validate evidence, a bottleneck that may be even more severe in realistic discovery workflows.

Takeaway. *Scientific discovery in ML is a full workflow, not just a proposal step. In a realistic compute-limited setting, current agents struggle with the broader judgment needed to choose informative experiments, allocate scarce trials, and turn feedback into evidence for scalable claims.*

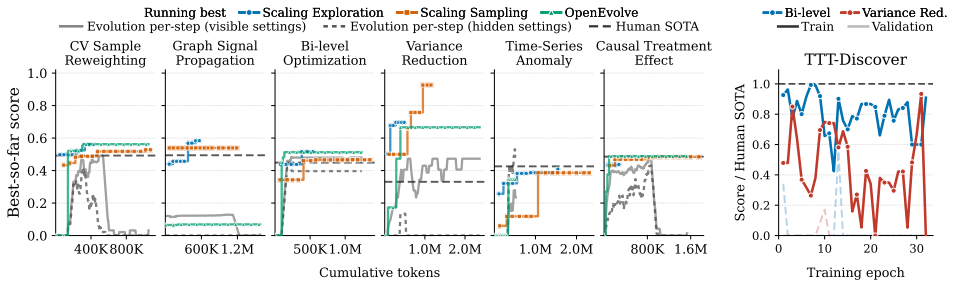

Figure 8. Test-time scaling. **Left:** running-best score vs. cumulative for the three inference-time setups. **Right:** TTT-Discover trained on two tasks, both hacking the visible settings.

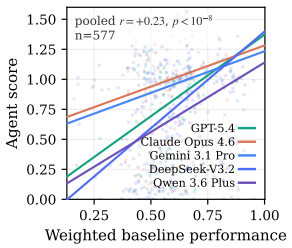

Figure 9. Similarity-weighted baseline performance vs. agent performance.

5.3. Context Engineering and Reasoning Patterns

We test whether additional context can benefit the agents. We add three settings: (1) **Web search**, where we equip the agents with a strong search tool based on Tavily[1]; (2) **Baseline ctx.**, which provides detailed derivations, key steps, and reasoning from the baseline papers; and (3) **Theory ctx.**, which provides background from relevant textbooks or theory-oriented literature.

Figure 7 shows that these interventions provide generally modest gains. Whether context helps depends on the model's own capability: stronger models can extract some benefit. For all models, even when gains appear, they are easily matched by ordinary iterative refinement. This suggests that the bottleneck is not access to knowledge, but the ability to turn knowledge into reasonable hypotheses.

> ***Takeaway.*** *The bottleneck is not missing knowledge, but using it: turning context into testable hypotheses, relevant evidence, and implementations that survive evaluation.*

5.4. Case Studies, Expert Assessment, and Error Analysis

Across six domain-author case studies that compare each agent edit with the same scaffold and a strong human baseline (Appendix F), expert assessment reveals dominant patterns: agents recombine ingredients drawn from the baselines they are shown and present the recombination as new, while truly novel components are rare and, when they appear, usually lack a stated reason to help. Per-model style differs: `GPT-5.4` reaches for the most structurally different ideas but tends to overclaim novelty; `Claude Opus 4.6` is the most disciplined, favoring careful tuning over architectural rewrites and producing the cleanest implementations; `Gemini 3.1 Pro` attempts the boldest changes but rarely backs them with explicit hypotheses; `DeepSeek-V3.2` and `Qwen-3.6 Plus` default to hyperparameter search dressed up as method discovery.

We further probe this pattern with a statistic. For each run we score code similarity to every baseline with 1/3-gram Jaccard, and form $q = \sum_i s_i \beta_i / \sum_i s_i$, the average baseline performance weighted by code similarity, where β_i is the baseline score. To pool across tasks, we normalize both q and agent score by the task's best baseline. Figure 9 plots these quantities. The pooled trend is significantly positive, and all five models follow the same direction. The per-model slope is significant for `DeepSeek-V3.2`, `Qwen-3.6 Plus`, and `GPT-5.4`, the three lowest-ranked models on the main leaderboard. This suggests a stratified failure mode that lower-performing models are more likely to mimic strong baselines than to explore new methods.

> ***Takeaway.*** *Current agents often produce combinations of the baselines they see. This pattern is especially clear for weaker models, whose performance tracks the baseline methods their code resembles.*

6. Conclusion and Future Work

We introduced MLS-Bench, a benchmark for evaluating whether AI systems can make reusable and scalable contributions to ML science. Across 140 tasks and 12 domains, current frontier agents remain far from reliably surpassing human-designed methods. MLS-Bench provides a controlled way to measure this gap as models and discovery methods advance.

Our analysis shows that the limitation of current agents lies not only in proposing better methods, but also in turning ideas into evidence. This evidence-building process requires a scientific discovery loop: deciding what to test, allocating limited trials, learning from feedback, and judging when a result supports a scalable claim. An agent can succeed on MLS-Bench only if it carries out this loop with sufficient rigor, not merely by proposing plausible methods.

MLS-Bench captures a more realistic scientific discovery regime than prior self-evolving settings, which often rely on cheap verifiers for large-scale sampling. In ML science and many real scientific problems, verification is expensive, multi-stage, delayed, and only partially observable. MLS-Bench points to the next core challenge for foundation models and agent harnesses: **scalable discovery under unscalable verification**.

[1] https://www.tavily.com/

Impact Statement

MLS-Bench is intended to make progress in automated ML science more measurable, comparable, and auditable. By exposing where current agents fail under controlled verification, it can steer the field away from demonstrations that conflate code editing, proxy optimization, and genuine method discovery. The benchmark uses controlled codebases, reproduced baselines, and bounded evaluation harnesses, and therefore does not directly deploy models in sensitive or human-facing settings. Its main risk is overinterpretation: a high score should not be treated as evidence of general scientific autonomy, and optimizing agents against any fixed benchmark can encourage proxy hacking if results are not checked independently. We therefore view MLS-Bench as one evaluation signal to be paired with domain expertise, independent replication, and careful validation before AI-discovered methods are used in real systems. While MLS-Bench takes the first step and reveals limitations of current agents, ML science is too broad and fast-moving for one benchmark to exhaust. Future work should explore evaluation designs that remain rigorous while giving agents more freedom to pursue more open-ended questions.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Anthropic. System card: Claude opus 4 & claude sonnet 4. https://www-cdn.anthropic.com/07b2a3f9902ee19fe39a36ca638e5ae987bc64dd.pdf, 2025.

Anthropic. Automated alignment researchers. https://www.anthropic.com/research/automated-alignment-researchers, 2026.

Appel, K. I. and Haken, W. The solution of the four-color-map problem. *Scientific American*, 237(4):108–121, 1977.

Assumpção, H., Ferreira, D., Campos, L., and Murai, F. Codeevolve: An open source evolutionary coding agent for algorithm discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2025.

AutoLab Team. Autolab: Can models begin to participate in the loops that drive scientific and engineering progress?, 2026. URL https://github.com/autolabhq/autolab.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Bragg, J., D'Arcy, M., Balepur, N., Bareket, D., Dalvi, B., Feldman, S., Haddad, D., Hwang, J. D., Jansen, P. A., Kishore, V., Majumder, B. P., Naik, A., Rahamimov, S., Richardson, K., Singh, A., Surana, H., Tiktinsky, A., Vasu, R., Wiener, G., Anastasiades, C., Candra, S., Dunkelberger, J., Emery, D., Evans, R., Hamada, M., Huff, R., Kinney, R., Latzke, M., Lochner, J., Lozano-Aguilera, R., Nguyen, C., Rao, S., Tanaka, A., Vlahos, B., Clark, P., Downey, D., Goldberg, Y., Sabharwal, A., and Weld, D. S. Astabench: Rigorous benchmarking of ai agents with a scientific research suite. *arXiv preprint arXiv:2510.21652*, 2025.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E. J., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *Neural Information Processing Systems*, 33:1877–1901, 2020.

Caballero, E., Gupta, K., Rish, I., and Krueger, D. Broken neural scaling laws. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sckjveqlCZ.

Chan, J. S., Chowdhury, N., Jaffe, O., Aung, J., Sherburn, D., Mays, E., Starace, G., Liu, K., Maksin, L., Patwardhan, T., Madry, A., and Weng, L. Mle-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025.

Chen, H., Xiong, M., Lu, Y., Han, W., Deng, A., He, Y., Wu, J., Li, Y., Liu, Y., and Hooi, B. Mlr-bench: Evaluating ai agents on open-ended machine learning research. *arXiv preprint arXiv:2505.19955*, 2025a.

Chen, J., Chen, W., Du, J., Hu, J., Jiang, Z., Jie, A., Jin, X., Jin, X., Li, C., Shi, W., Wang, Z., Wang, M., Wei, C., Wei, S., Xin, H., Yang, F., Gao, W., Yuan, Z., Zhan, T., Zheng, Z., Zhou, T., and Zhu, T. H. Seed-prover 1.5: Mastering undergraduate-level theorem proving via learning from experience. *arXiv preprint arXiv:2512.17260*, 2025b.

Chen, J., Xu, X., Wei, H., Chen, C., and Zhao, B. Swe-ci: Evaluating agent capabilities in maintaining

codebases via continuous integration. *arXiv preprint arXiv:2603.03823*, 2026a.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, T., Zheng, L., Yan, E. Q., Jiang, Z., Moreau, T., Ceze, L., Guestrin, C., and Krishnamurthy, A. Learning to optimize tensor programs. 2018.

Chen, T., Ye, Z., Xu, B., Ye, Z., Liu, T., Hassani, A., Chen, T., Kerr, A., Wu, H., Xu, Y., Chen, Y.-J., Chen, H., Kane, A., Krashinsky, R., Liu, M.-Y., Grover, V., Ceze, L., Bringmann, R. A., Tran, J., Liu, W., Xie, F., Lightstone, M., and Shi, H. Avo: Agentic variation operators for autonomous evolutionary search. *arXiv preprint arXiv:2603.24517*, 2026b.

Chen, W., Yang, X., Yang, X., Sha, T., Li, Q., Wang, Z., Xian, B., Kong, F., Liu, W., and Bian, J. Agent^2 rl-bench: Can llm agents engineer agentic rl post-training? *arXiv preprint arXiv:2604.10547*, 2026c.

Chen, Z., Chen, S., Ning, Y., Zhang, Q., Wang, B., Yu, B., Li, Y., Liao, Z., Wei, C., Lu, Z., Dey, V., Xue, M., Baker, F. N., Burns, B., Adu-Ampratwum, D., Huang, X., Ning, X., Gao, S., Su, Y., and Sun, H. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations*, 2025c.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhu, B., Zhang, H., Jordan, M. I., Gonzalez, J. E., and Stoica, I. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, pp. 8359–8388, 2024.

Coley, C. W., Thomas, D. A., Lummiss, J. A. M., Jaworski, J. N., Breen, C., Schultz, V., Hart, T., Fishman, J., Rogers, L., Gao, H., Hicklin, R. W., Plehiers, P., Byington, J., Piotti, J. S., Green, W. H., Hart, A. J., Jamison, T. F., and Jensen, K. F. A robotic platform for flow synthesis of organic compounds informed by ai planning. *Science*, 365(6453), 2019.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Everett, K. E., Xiao, L., Wortsman, M., Alemi, A. A., Novak, R., Liu, P. J., Gur, I., Sohl-Dickstein, J., Kaelbling, L. P., Lee, J., and Pennington, J. Scaling exponents across parameterizations and optimizers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12666–12700. PMLR, 2024. URL https://proceedings.mlr.press/v235/everett24a.html.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135, 2017.

Garikaparthi, A., Patwardhan, M., and Cohan, A. Researchgym: Evaluating language model agents on real-world ai research. *arXiv preprint arXiv:2602.15112*, 2026.

Gonzalez, S. and Miikkulainen, R. Improved training speed, accuracy, and data utilization through loss function optimization. In *IEEE Congress on Evolutionary Computation*, pp. 1–8, 2020.

Gottweis, J., Weng, W.-H., Daryin, A. N., Tu, T., Palepu, A., Sirkovic, P., Myaskovsky, A., Weissenberger, F., Rong, K., Tanno, R., Saab, K., Popovici, D., Blum, J., Zhang, F., Chou, K., Hassidim, A., Gokturk, B., Vahdat, A., Kohli, P., Matias, Y., Carroll, A., Kulkarni, K., Tomasev, N., Guan, Y., Dhillon, V., Vaishnav, E. D., Lee, B., Costa, T. R. D., Penadés, J. R., Peltz, G., Xu, Y., Pawlosky, A., Karthikesalingam, A., and Natarajan, V. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C.,

Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, January 2026. URL https://github.com/harbor-framework/harbor.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. doi: 10.48550/arXiv.2203.15556. URL https://arxiv.org/abs/2203.15556.

Huaizheng Zhang*, Yizheng Huang*, L. Z. Mle-agent: Your intelligent companion for seamless ai engineering and research, 2024. https://github.com/MLSysOps/MLE-agent.

Huang, Q., Vora, J., Liang, P., and Leskovec, J. Mlagentbench: Evaluating language agents on machine learning experimentation. In *Forty-first International Conference on Machine Learning*, pp. 20271–20309, 2024.

Hubert, T., Mehta, R., Sartran, L., Horváth, M. Z., Žužić, G., Wieser, E., Huang, A., Schrittwieser, J., Schroecker, Y., Masoom, H., et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, pp. 1–3, 2025.

Hutter, F., Kotthoff, L., and Vanschoren, J. Automated machine learning - methods, systems, challenges. 2019.

Imajuku, Y., Horie, K., Iwata, Y., Aoki, K., Takahashi, N., and Akiba, T. Ale-bench: A benchmark for long-horizon objective-driven algorithm engineering. *arXiv preprint arXiv:2506.09050*, 2025.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Köhl, S., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zieliński, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S. W., Silver, D., Vinyals, O., Senior, A., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kaddour, J., Key, O., Nawrot, P., Minervini, P., and Kusner, M. J. No train no gain: Revisiting efficient training algorithms for transformer-based language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.

neurips.cc/paper_files/paper/2023/hash/51f3d6252706100325ddc435ba0ade0e-Abstract-Conference.html.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karpathy, A. autoresearch. https://github.com/karpathy/autoresearch, 2025.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krenn, M., Pollice, R., Guo, S. Y., Aldeghi, M., Cervera-Lierta, A., Friederich, P., dos Passos Gomes, G., Häse, F., Jinich, A., Nigam, A., Yao, Z., and Aspuru-Guzik, A. On scientific understanding with artificial intelligence. *Nature Reviews Physics*, 4(12):761–769, 2022.

Lam, R., Álvaro Sánchez-González, Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

Lange, R. T., Imajuku, Y., and Cetin, E. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Li, M., Kudugunta, S., and Zettlemoyer, L. (mis)fitting scaling laws: A survey of scaling law fitting techniques in deep learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xI71dsS3o4.

Liang, J., Lyu, Z., Liu, Z., Chen, X., Nie, P., Zou, K., and Chen, W. Swe-next: Scalable real-world software engineering tasks for agents. *arXiv preprint arXiv:2603.20691*, 2026.

Lin, Y., Tang, S., Lyu, B., Wu, J., Lin, H., Yang, K., Li, J., Xia, M., Chen, D., Arora, S., and Jin, C. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025a.

Lin, Y., Tang, S., Lyu, B., Yang, Z., Chung, J.-H., Zhao, H., Jiang, L., Geng, Y., Ge, J., Sun, J., Wu, J., Gesi, J., Lu, X., Acuna, D., Yang, K., Lin, H., Choi, Y., Chen, D., Arora, S., and Jin, C. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025b.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *7th International Conference on Learning Representations*, 2019.

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a.

Liu, Y., Yang, Z., Xie, T., Ni, J., Gao, B., Li, Y., Tang, S., Ouyang, W., Cambria, E., and Zhou, D. Researchbench: Benchmarking llms in scientific discovery via inspiration-based task decomposition. *arXiv preprint arXiv:2503.21248*, 2025b.

Lu, C., Lu, C., Lange, R. T., Foerster, J. N., Clune, J., and Ha, D. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Lupidi, A. M., Gauri, B., Foster, T., Omari, B. A., Magka, D., Pepe, A., Audran-Reiss, A., Aghamelu, M., Baldwin, N. M., Cipolina-Kun, L., Gagnon-Audet, J.-C., Leow, C. H., Lefdal, S., Mossalam, H., Moudgil, A., Nazir, S., Tewolde, E., Urrego, I., Armengol-Estapé, J., Budhiraja, A., Chaurasia, G., Charnalia, A., Dunfield, D., Hambardzumyan, K., Izcovich, D., Josifoski, M., Mediratta, I., Niu, K., Pathak, P., Shvartsman, M., Toledo, E., Protopopov, A., Raileanu, R., Miller, A. H., Shavrina, T., Foerster, J. N., and Bachrach, Y. Airs-bench: a suite of tasks for frontier ai research science agents. *arXiv preprint arXiv:2602.06855*, 2026.

Majumder, B. P., Surana, H., Agarwal, D., Mishra, B. D., Meena, A., Prakhar, A., Vora, T., Khot, T., Sabharwal, A., and Clark, P. Discoverybench: Towards data-driven discovery with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mang, Q., Chai, W., Li, Z., Mao, H., Zhou, S., Du, A., Li, H., Liu, S., Chen, E., Wang, Y., Chu, X., Cheng, Z., Xu, Y., Xia, T., Wang, Z., Shi, T., Yao, J., Zhao, Y., Zhang, Q., Ruan, C., Shen, Z., Liu, K., He, R., Xing, D., Li, Z., Zeng, Z., Jiang, Y., Cheng, L., Zhao, Z., Sun, Y., Zheng, W., Zhang, M., Ji, R., Tu, X., Zheng, Z., Chen, Z., Zhou, K., Wang, Z., Chen, J., Korolova, A., Henderson, P., Viswanath, P., Ganesh, V., Xie, S., Liu, Z., Song, D., Min, S., Stoica, I., Gonzalez, J. E., Shang, J., and Cheung, A. Frontiercs: Evolving challenges for evolving intelligence. *arXiv preprint arXiv:2512.15699*, 2025.

Mankowitz, D. J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J.-B., Ahern, A., Köppe, T., Millikin, K., Gaffney, S., Elster, S., Broshear, J., Gamble, C., Milan, K., Tung, R., Hwang, M., Cemgil, A. T., Barekatain, M., Li, Y., Mandhane, A., Hubert, T., Schrittwieser, J., Hassabis, D., Kohli, P., Riedmiller, M. A., Vinyals, O., and Silver, D. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023.

Merchant, A., Batzner, S. L., Schoenholz, S. S., Aykol, M., Cheon, G., and Cubuk, E. D. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2024.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nova, A., et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mündler, N., Müller, M. N., He, J., and Vechev, M. Swt-bench: Testing and validating real-world bug-fixes with code agents. *Advances in Neural Information Processing Systems*, 37:81857–81887, 2024.

Nathani, D., Madaan, L., Roberts, N., Bashlykov, N., Menon, A., Moens, V., Budhiraja, A., Magka, D., Vorotilov, V., Chaurasia, G., Hupkes, D., Cabral, R. S., Shavrina, T., Foerster, J. N., Bachrach, Y., Wang, W. Y., and Raileanu, R. Mlgym: A new framework and benchmark for advancing ai research agents. *arXiv preprint arXiv:2502.14499*, 2025.

Novikov, A., Vu, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Openai o3 and o4-mini system card. https://openai.com/index/o3-o4-mini-system-card/, 2025. System card, published April 16, 2025.

Ouyang, A., Guo, S., Arora, S., Zhang, A. L., Hu, W., Ré, C., and Mirhoseini, A. Kernelbench: Can llms write efficient gpu kernels? In *Forty-second International Conference on Machine Learning*, 2025.

Panapitiya, G., Saldanha, E., Job, H., and Hess, O. Autolabs: Cognitive multi-agent systems with self-correction for autonomous chemical experimentation. *arXiv preprint arXiv:2509.25651*, 2025.

Panfilov, A., Romov, P., Shilov, I., de Montjoye, Y.-A., Geiping, J., and Andriushchenko, M. Claudini: Autoresearch discovers state-of-the-art adversarial attack algorithms for llms. *arXiv preprint arXiv:2603.24511*, 2026.

Panigrahi, S. S., Videnovic, J., and Brbic, M. Heurekabench: A benchmarking framework for ai co-scientist. *arXiv preprint arXiv:2601.01678*, 2026.

Phan, P., Agarwal, D., Srinivas, K., Samulowitz, H., Kapanipathi, P., and McCallum, A. Migrate: Mixed-policy grpo for adaptation at test-time. *arXiv preprint arXiv:2508.08641*, 2025.

Press, O., Amos, B., Zhao, H., Wu, Y., Ainsworth, S. K., Krupke, D., Kidger, P., Sajed, T., Stellato, B., Park, J., Bosch, N., Meril, E., Steppi, A., Zharmagambetov, A., Zhang, F., Perez-Pineiro, D., Mercurio, A., Zhan, N., Abramovich, T., Lieret, K., Zhang, H., Huang, S., Bethge, M., and Press, O. Algotune: Can language models speed up general-purpose numerical programs? *arXiv preprint arXiv:2507.15887*, 2025.

Qiang, R., Zhuang, Y., Li, Y., K, D. S. V., Zhang, R., Li, C., Wong, I. S.-H., Yang, S., Liang, P., Zhang, C., and Dai, B. Mle-dojo: Interactive environments for empowering llm agents in machine learning engineering. *arXiv preprint arXiv:2505.07782*, 2025.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Rank, B., Bhatnagar, H., Prabhu, A., Eisenberg, S., Nguyen, K., Bethge, M., and Andriushchenko, M. Posttrainbench: Can llm agents automate llm post-training? *arXiv preprint arXiv:2603.08640*, 2026.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/adc98a266f45005c403b8311ca7e8bd7-Abstract-Conference.html.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach

themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Schmidgall, S., Su, Y., Wang, Z., Sun, X., Wu, J., Yu, X., Liu, J., Moor, M., Liu, Z., and Barsoum, E. Agent laboratory: Using llm agents as research assistants. In *Findings of the Association for Computational Linguistics*, pp. 5977–6043, 2025.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, R., Asai, A., Shen, S. Z., Ivison, H., Kishore, V., Zhuo, J., Zhao, X., Park, M., Finlayson, S. G., Sontag, D. A., Murray, T., Min, S., Dasigi, P., Soldaini, L., Brahman, F., tau Yih, W., Wu, T., Zettlemoyer, L., Kim, Y., Hajishirzi, H., and Koh, P. W. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

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

Si, C., Yang, Z., Choi, Y., Candès, E., Yang, D., and Hashimoto, T. Towards execution-grounded automated ai research. *arXiv preprint arXiv:2601.14525*, 2026.

Simon, H. A. The science of design: Creating the artificial. *Design issues*, pp. 67–82, 1988.

Starace, G., Jaffe, O., Sherburn, D., Aung, J., Chan, J. S., Maksin, L., Dias, R., Mays, E., Kinsella, B., Thompson, W., Heidecke, J., Glaese, A., and Patwardhan, T. Paperbench: Evaluating ai's ability to replicate ai research. In *Forty-second International Conference on Machine Learning*, 2025.

Surina, A., Mansouri, A., Quaedvlieg, L., Seddas, A., Viazovska, M., Abbe, E., and Gulcehre, C. Algorithm discovery with llms: Evolutionary search meets reinforcement learning. *arXiv preprint arXiv:2504.05108*, 2025.

Swanson, K., Wu, W., Bulaong, N. L., Pak, J. E., and Zou, J. The virtual lab: Ai agents design new sars-cov-2 nanobodies with experimental validation. 2024.

Tang, J., Xia, L., Li, Z., and Huang, C. Ai-researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*, 2025.

Tang, X., Liu, Y., Cai, Z., Shao, Y., Lu, J., Zhang, Y., Deng, Z., Hu, H., An, K., Huang, R., Si, S., Chen, S., Zhao, H., Chen, L., Wang, Y., Liu, T., Jiang, Z., Chang, B., Fang, Y., Qin, Y., Zhou, W., Zhao, Y., Cohan, A., and Gerstein, M. Ml-bench: Evaluating large language models and agents for machine learning tasks on repository-level code. *arXiv preprint arXiv:2311.09835*, 2023.

Team, G. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Team, K. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025a.

Team, K. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025b.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Udrescu, S. and Tegmark, M. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631–eaay2631, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vincenti, W. G. et al. *What engineers know and how they know it*, volume 141. Baltimore: Johns Hopkins University Press, 1990.

Vitvitskyi, A., Boratko, M., Grcic, M., Pascanu, R., Shah, D., and Velickovic, P. Mining generalizable activation functions. *arXiv preprint arXiv:2602.05688*, 2026.

Wang, H., Unsal, M., Lin, X., Baksys, M., Liu, J., Santos, M. D., Sung, F., Vinyes, M., Ying, Z., Zhu, Z., Lu, J., de Saxcé, H., Bailey, B., Song, C., Xiao, C., Zhang, D., Zhang, E., Pu, F., Zhu, H., Liu, J., Bayer, J., Michel, J., Yu, L., Dreyfus-Schmidt, L., Tunstall, L., Pagani, L., Machado, M., Bourigault, P., Wang, R., Polu, S., Barroyer, T., Li, W.-D., Niu, Y., Fleureau, Y., Hu, Y., Yu, Z., Wang, Z., Yang, Z., Liu, Z., and Li, J. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025a.

Wang, Y., Su, S.-R., Zeng, Z., Xu, E., Ren, L., Yang, X., Huang, Z., He, X., Ma, L., Peng, B., Cheng, H., He, P.,

Chen, W., Wang, S., Du, S. S., and Shen, Y. Thetaevolve: Test-time learning on open problems. *arXiv preprint arXiv:2511.23473*, 2025b.

Wang, Y., Zhang, Y., Wu, Y., Lu, L., Le Nguyen, P., Wang, X., and Nguyen, C.-T. Mlalgo-bench: Can machines implement machine learning algorithms? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14298–14329, 2025c.

Wang, Z., Bai, F., Luo, Z., Su, J., Sun, K., Yu, X., Liu, J., Zhou, K., Cardie, C., Dredze, M., Xing, E. P., and Hu, Z. Fire-bench: Evaluating agents on the rediscovery of scientific insights. *arXiv preprint arXiv:2602.02905*, 2026.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Torres, S. V., Lauko, A., Bortoli, V. D., Émile Mathieu, Ovchinnikov, S., Barzilay, R., Jaakkola, T., DiMaio, F., Baek, M., and Baker, D. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=yzkSU5zdwD.

Wei, J., Sun, Z., Papay, S., McKinney, S., Han, J., Fulford, I., Chung, H. W., Passos, A. T., Fedus, W., and Glaese, A. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Wen, K., Hall, D., Ma, T., and Liang, P. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025. doi: 10.48550/arXiv.2509.02046. URL https://arxiv.org/abs/2509.02046.

Wijk, H., Lin, T. R., Becker, J., Jawhar, S., Parikh, N., Broadley, T., Chan, L., Chen, M., Clymer, J., Dhyani, J., Ericheva, E., Garcia, K., Goodrich, B., Jurkovic, N., Kinniment, M., Lajko, A., Nix, S., Sato, L. J. K., Saunders, W., Taran, M., West, B., and Barnes, E. Re-bench: Evaluating frontier ai r&d capabilities of language model agents against human experts. In *Forty-second International Conference on Machine Learning*, 2025.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J. N., Clune, J., and Ha, D. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B. X., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Feng, H., Ge, H., Wei, H., Huan, L., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Yang, F., Yang, S., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yang, G., Hu, E. J., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., and Gao, J. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/8df7c2e3c3c3be098ef7b382bd2c37ba-Abstract.html.

Yang, S., He-Yueya, J., and Liang, P. Reinforcement learning for machine learning engineering agents. *arXiv preprint arXiv:2509.01684*, 2025b.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Ye, H., Lin, H., Tang, J., Luo, Y., Yang, C., Su, C., Thapa, R., Yang, R., Liu, R., Li, Z., Gao, C., Ding, D., He, G., Zhang, M., Sun, L., Wang, W., Zhong, Y., Shen, Z., He, D., Ma, J., Ermon, S., Li, T., Chu, X., Zou, J., and Xu, Y. Evaluation-driven scaling for scientific discovery. *arXiv preprint arXiv:2604.19341*, 2026. URL https://arxiv.org/abs/2604.19341.

Yuksekgonul, M., Koceja, D. M., Li, X., Bianchi, F., McCaleb, J., Wang, X., Kautz, J., Choi, Y., Zou, J., Guestrin, C., and Sun, Y. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

Zhang, B. and Sennrich, R. Root mean square layer normalization. *Neural Information Processing Systems*, 32: 12360–12371, 2019.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations*, 2017.

Zuo, Y., Zhang, K., Sheng, L., Qu, S., Cui, G., Zhu, X., Li, H., Zhang, Y., Long, X., Hua, E., et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A. Full Task Catalog

Table 4 lists the full MLS-Bench task catalog, grouped by research area, with each task's research question, external package(s), baselines, and evaluation settings.

Table 4. The full MLS-Bench task catalog grouped by research area. Each row gives the formal task name, a one-sentence research question, the external package(s) that supply the training and evaluation pipeline (`author/repo` for upstream packages, *custom* for in-house scaffolds), the registered baselines, and the model/dataset evaluation settings.

Name	Description	External Package(s)	Baselines	Evaluation Settings
Language Models (LM)				
LLM Agent Tool-Use Reasoning Strategy	Studies how tool-use search, backtracking, and stopping policies affect answer validity and query efficiency.	`zhichengg/StableToolBench`	Greedy Chain (CoT) DFS with LLM Ranking DFSDT	StableToolBench I1-instruction 50q / deepseek-chat StableToolBench I1-instruction 50q / qwen2.5-72b-instruct StableToolBench I1-instruction 50q / qwen2.5-7b-instruct
Masked Diffusion LM: Demasking Strategy	Studies how demasking schedules, position selection, and token assignment affect diffusion language-model quality and decoding efficiency.	`ML-GSAI/LLaDA`	Top-K Margin Confidence Greedy KLASS	LLaDA / MATH-500 LLaDA / HumanEval Dream / C4 prefix continuation
Autoregressive Attention Mechanism	Studies how self-attention computation and positional handling affect autoregressive pretraining loss and downstream accuracy.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	QK-Norm RoPE RoPE + QK-Norm	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Low-Bit Linear Pretraining Layer	Studies how low-bit linear layers and quantization functions affect pretraining loss under discrete weight constraints.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	Binary Sign (BitNet) Ternary 1.58-bit (BitNet b1.58) INT2 Uniform	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Autoregressive Embedding Strategy	Studies how token embeddings, position embeddings, value embeddings, and weight tying affect autoregressive pretraining loss and downstream accuracy.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	Untied Embeddings Value Embeddings Bigram Hash Embeddings	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Subquadratic Attention Mechanism	Studies whether linear or subquadratic attention can reduce autoregressive validation loss while preserving downstream performance.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	RetNet DeltaNet GLA	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Autoregressive Pretraining Loss	Studies how alternative next-token training losses affect autoregressive validation cross-entropy.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	Label Smoothing Softcap Cross-Entropy Z-Loss	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Pretraining Learning-Rate Schedule	Studies how warmup, decay shape, and schedule horizon affect autoregressive pretraining validation loss.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	WSD (Warmup-Stable-Decay) Trapezoidal WSD with Inverse-Sqrt Decay	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Transformer Feed-Forward Block	Studies how activation, gating, and expansion choices in the feed-forward sublayer affect language-model validation loss.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	ReLU-Squared SwiGLU GeGLU	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Normalization and Block Layout	Studies how normalization placement, affine behavior, and transformer block layout affect pretraining stability and validation loss.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	RMSNorm RMSNorm + Sandwich-Norm RMSNorm (Parallel Block)	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Pretraining Optimizer Design	Studies how optimizer choice, parameter grouping, and schedule coupling affect autoregressive pretraining validation loss.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	AdamW + Nesterov Lion Muon	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Transformer Residual Stream Strategy	Studies how residual connections and information flow across transformer layers affect validation loss, perplexity, and accuracy metrics.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	Vanilla (Pre-LN) ProRes Learned Scaling Block Attention Residuals	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
Reasoning RL Advantage Estimation	Studies how advantage estimates for online language-model reinforcement learning affect mathematical reasoning accuracy.	`volcengine/verl`	GRPO Dr. GRPO Reinforce++ Baseline	GSM8K MATH-500 AMC
Reasoning RL Importance-Sampling Granularity	Studies how importance-sampling ratio granularity and clipping affect online language-model reinforcement learning for reasoning.	`volcengine/verl`	Token-Level (Vanilla PPO) Sequence-Level (GSPO) First-K Tokens	GSM8K MATH-500 AMC
Actor Divergence Estimator for Reasoning RL	Studies how per-token actor KL estimation controls reference-policy drift while preserving reasoning accuracy during online RL.	`volcengine/verl`	K1 (Unbiased Log-Ratio) K2 (Squared Log-Ratio) K3 (Low-Variance KL) Absolute Log-Ratio	GSM8K MATH-500 AMC
Pre-Advantage Reward Normalization	Studies how reward normalization before advantage estimation affects reasoning accuracy in online language-model RL.	`volcengine/verl`	Outcome-Only (Raw) Group-Std Normalization Batch-Std Whitening Length-Aware Normalization	GSM8K MATH-500 AMC
Symbolic Scaling-Law Discovery	Studies how symbolic functional forms and group-specific coefficients capture held-out scaling behavior.	`trevorstephens/gplearn`	Human Exact Form SLDAgent-Style Kernel Ridge Regression XGBoost	SLDBench Vocabulary Scaling SLDBench LR x Batch-Size Scaling SLDBench Data-Constrained Scaling

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Language-Agent Collaboration Topology	Studies how deterministic collaboration topology affects multi-agent code-generation quality and execution success.	`OpenBMB/ChatDev`	Chain Star Layered	HumanEval-33 (deepseek-chat, 4 agents) HumanEval-33 (qwen2.5-72b-instruct, 4 agents) SRDD-20 (deepseek-chat, 4 agents)
Robotics (ROB)				
Latent World-Model Planner	Studies how goal-conditioned planning should exploit a fixed latent world model to improve navigation success.	`facebookresearch/eb_jepa`	Random CEM MPPI iCEM	Two Rooms (Horizon 30) Two Rooms (Horizon 60) Two Rooms (Horizon 90)
Temporal Latent Prediction Loss	Studies how latent prediction objectives affect multi-step video representation quality.	`facebookresearch/eb_jepa`	MSE Smooth L1 Cosine	Moving MNIST AP (small: henc=16, dstc=8, hpre=16) Moving MNIST AP (base: henc=32, dstc=16, hpre=32) Moving MNIST AP (large: henc=64, dstc=32, hpre=64)
Anti-Collapse Representation Regularizer	Studies how self-supervised regularization prevents representation collapse and improves linear-probe accuracy.	`facebookresearch/eb_jepa`	Naive VICReg SigReg Barlow Twins	ResNet-18 Probe ResNet-34 Probe ResNet-50 Probe
Diffusion Guidance for Robot Trajectory Planning	Studies guidance mechanisms for a fixed trajectory-level diffusion planner on D4RL MuJoCo, optimizing normalized score across hopper-medium-v2, walker2d-medium-v2, and halfcheetah-medium-v2.	`CleanDiffuserTeam/CleanDiffuser`	Diffuser (Classifier Guidance) Classifier-Free Guidance No Guidance Decision Diffuser	D4RL Hopper-Medium-v2 D4RL Walker2d-Medium-v2 D4RL HalfCheetah-Medium-v2
Diffusion Policy Learning for Robot Control	Studies how diffusion policy training, value guidance, and action generation affect robot-control episode reward.	`CleanDiffuserTeam/CleanDiffuser`	DQL (Diffusion Q-Learning) IDQL Diffusion Policy	D4RL Hopper-Medium-v2 D4RL Walker2d-Medium-v2 D4RL HalfCheetah-Medium-v2
Efficient Diffusion Sampling for Robot Actions	Studies how solver choice and sampling_steps affect DQL-style diffusion-policy normalized score at low NFE on D4RL MuJoCo.	`CleanDiffuserTeam/CleanDiffuser`	DDPM (100-Step Ancestral Sampling) DDIM (20-Step Deterministic Sampling) DPM-Solver++ 2M (10-Step)	D4RL Hopper-Medium-v2 D4RL Walker2d-Medium-v2 D4RL HalfCheetah-Medium-v2
Humanoid Transfer Policy Learning	Studies how actor-critic architecture, policy optimization, and rollout processing affect humanoid command-following transfer.	`roboterax/humanoid-gym`	Default PPO PPO with Adaptive KL PPO with LayerNorm	RobotEra XBot-L Training RobotEra XBot-L / Diverse Commands RobotEra XBot-L / Forward-Only RobotEra XBot-L / High Speed
Behavioral Cloning Loss for Manipulation	Studies how imitation-learning loss design affects rollout success for low-dimensional robot manipulation tasks.	`ARISE-Initiative/robomimic`	NLL with Entropy Weighted NLL Default (NLL)	Tool Hang (PH) Can (PH) Square (PH)
Offline Value Loss for Manipulation	Studies how asymmetric value regression loss design affects offline robot manipulation policy success.	`ARISE-Initiative/robomimic`	Quantile Regression Huber Pinball Default (Expectile)	Tool Hang (PH) Can (PH) Square (PH)

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Observation Fusion Encoder for Imitation Learning	Designs a multimodal robot state encoder for behavioral cloning to improve rollout success rate on manipulation tasks.	`ARISE-Initiative/robomimic`	Attention Fusion Gated Fusion Default (Concatenation)	Tool Hang (PH) Can (PH) Square (PH)
Trajectory Optimization for Model-Based Planning	An online planning algorithm selects actions through learned-world-model trajectory optimization to improve episode reward.	`nicklashansen/tdmpc2`	CEM iCEM MPPI	Walker Walk Cheetah Run Cartpole Swingup
Latent Representation Normalization for Model-Based RL	Designs latent-state normalization for the TD-MPC2 encoder and dynamics world-model networks, evaluated by DMControl episode reward.	`nicklashansen/tdmpc2`	SimNorm L2 normalization RMSNorm Identity (no normalization)	DMControl walker-walk DMControl cheetah-run DMControl cartpole-swingup
Vision & Generation (V&G)				
3D Gaussian Splatting Densification Strategy Design	Designs a 3D Gaussian Splatting densification strategy controlling clone, split, prune, reset, relocation, and sample-add behavior to improve held-out novel-view quality on Mip-NeRF 360 scenes.	`nerfstudio-project/gsplat`	Original 3DGS densification AbsGS + Taming-3DGS + New Split EDC-TamingGS-Abs	Mip-NeRF 360 garden (8x, best PSNR) Mip-NeRF 360 bicycle (8x, best PSNR) Mip-NeRF 360 bonsai (8x, best PSNR) Mip-NeRF 360 stump (8x, best PSNR)
3D Gaussian Splatting Regularizer Design	Designs a scalar regularizer added to the 3DGS photometric loss during 30k-step Mip-NeRF 360 reconstruction, evaluated on held-out novel views and scored by best PSNR.	`nerfstudio-project/gsplat`	No regularization Scale + opacity L1 Effective-rank + scale/opacity L1	Mip-NeRF 360 garden (8x, best PSNR) Mip-NeRF 360 bicycle (8x, best PSNR) Mip-NeRF 360 bonsai (8x, best PSNR) Mip-NeRF 360 stump (8x, best PSNR)
Custom Sampler for Diffusion Bridge Models	Designs a low-NFE sampler for Diffusion Bridge Models on image-to-image translation, ImageNet center-inpainting, and DIODE depth, evaluated by FID at NFE=5.	`thu-ml/DiffusionBridge`	DBIM DBIM-HO (high-order) DDBM (50 NFE reference) ECSI	Edges2Handbags / e2h (FID, NFE=5) ImageNet center-inpaint (FID, NFE=5) DIODE depth (FID, NFE=5)
Time Scheduler for Diffusion Bridge Models (NFE=5)	Designs a monotone low-step time schedule for Diffusion Bridge Models, evaluated by FID on Edges2Handbags, ImageNet center-inpainting, and DIODE depth at NFE=5.	`thu-ml/DiffusionBridge`	Karras EDM (rho=7) Uniform (linear) Cosine (Nichol-Dhariwal) Log-linear (geometric)	Edges2Handbags / e2h (FID, NFE=5) ImageNet center-inpaint (FID, NFE=5) DIODE depth (FID, NFE=5)
Diffusion Model Architecture Design	Design a denoising UNet backbone for unconditional CIFAR-10 DDPM training, optimizing best FID with fixed epsilon prediction and 50-step DDIM sampling.	`huggingface/diffusers`	Standard DDPM U-Net Full-Attention U-Net No-Attention U-Net	CIFAR-10 DDPM Small CIFAR-10 DDPM Medium CIFAR-10 DDPM Large
Diffusion Model: Classifier-Free Guidance Optimization	Design a classifier-free guidance method for Stable Diffusion text-to-image generation across SD v1.5, Stable Diffusion 2 Base, and Stable Diffusion XL; evaluation generates COCO-caption images and official scoring uses per-model FID.	`CFGpp-diffusion/CFGpp`	Standard CFG CFG++ Zero-Init CFG++	Stable Diffusion v1.5 / COCO captions / NFE=10 Stable Diffusion 2 Base / COCO captions / NFE=10 Stable Diffusion XL Base 1.0 / COCO captions / NFE=10

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Class-Conditional Diffusion: Conditioning Injection Methods	Design class-conditioning injection for a CIFAR-10 class-conditional UNet2DModel/DDPM, optimizing best FID with 50-step DDIM sampling.	`huggingface/diffusers`	Concat-FiLM Cross-Attention AdaLN-Zero	CIFAR-10 Class-Conditional Small UNet2DModel CIFAR-10 Class-Conditional Medium UNet2DModel CIFAR-10 Class-Conditional Large UNet2DModel
Diffusion Model: Sampler Efficiency Optimization	Design a Stable Diffusion sampler update rule for COCO-caption text-to-image generation at a fixed NFE=20 budget; official scoring uses per-model FID.	`CFGpp-diffusion/CFGpp`	DDIM DPM++ 3M DPM++ 2S	Stable Diffusion v1.5 / COCO captions / NFE=20 Stable Diffusion 2 Base / COCO captions / NFE=20 Stable Diffusion XL Base 1.0 / COCO captions / NFE=20
Diffusion Prediction Parameterization	Design a prediction target and consistent x0 inversion for unconditional CIFAR-10 UNet2DModel diffusion, optimizing best FID with 50-step DDIM sampling.	`huggingface/diffusers`	Epsilon Prediction V-Prediction X0 Prediction	CIFAR-10 Unconditional Small UNet2DModel CIFAR-10 Unconditional Medium UNet2DModel CIFAR-10 Unconditional Large UNet2DModel
Flow Map with Perceptual Loss	Studies whether auxiliary perceptual losses on denoised images improve CIFAR-10 FID for MeanFlow flow-map training with DiT backbones.	`snap-research/alphaflow`	Pure MSE Velocity MSE + Charbonnier + LPIPS + Gradient + Multiscale MSE + LPIPS + Gradient + Multiscale + FFT	CIFAR-10 Small DiT CIFAR-10 Medium DiT CIFAR-10 Large DiT
VAE Loss Function Design for Image Reconstruction	Studies how VAE loss components affect CIFAR-10 AutoencoderKL reconstruction quality, scored primarily by rFID on the full test set.	`huggingface/diffusers`	L1 + KL L1 + LPIPS + KL L1 + LPIPS + KL + PatchGAN	CIFAR-10 AutoencoderKL Small CIFAR-10 AutoencoderKL Medium CIFAR-10 AutoencoderKL Large
Reinforcement Learning (RL)				
Cooperative MARL Centralized Critic Architecture for MAPPO	Studies centralized critic architectures for MAPPO on SMACLite cooperative MARL maps, scored by greedy-policy test win rate and return.	`uoe-agents/epymarl`	IPPO Decentralized Critic MAPPO Centralized Critic MAT-Style Attention Critic	SMACLite MMM (10-agent heterogeneous) SMACLite 2s3z (5-agent heterogeneous) SMACLite 3s5z (8-agent heterogeneous)
Meta-RL: Context Encoder for PEARL Task Inference	Studies PEARL context encoders that map transition tuples to latent task representations for fast adaptation, evaluated by meta_test_return after 20 meta-training iterations.	`katerakelly/oyster`	PEARL MLP Context Encoder PEARL Recurrent Context Encoder PEARL Attention Context Encoder	Half-Cheetah Velocity (30 train/10 test tasks) Sparse Point Robot (40 train/10 test tasks) Point Robot (40 train/10 test tasks)
Meta-RL Algorithm Design	Studies complete meta-RL algorithm design across task inference, policy conditioning, and meta-training, scored by meta_test_return on held-out tasks after the fixed short-budget protocol.	`katerakelly/oyster`	PEARL FOCAL VariBAD	Half-Cheetah Velocity (30 train/10 test tasks) Sparse Point Robot (40 train/10 test tasks) Point Robot (40 train/10 test tasks)
Intrinsic Exploration for Sparse Rewards	Studies how intrinsic rewards and advantage mixing affect exploration and return in sparse-reward Atari environments.	`vwxyzjn/cleanrl`	PPO RND ICM	Tutankham-v5 Frostbite-v5 PrivateEye-v5

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Offline Dexterous Manipulation from Narrow Demonstrations	Studies how offline RL algorithms learn dexterous manipulation from narrow human demonstration datasets.	`corl-team/CORL`	IQL AWAC ReBRAC	Pen-Human-v1 Hammer-Human-v1 Door-Cloned-v1
Q-Overestimation Suppression for Offline Continuous Control	Studies how offline continuous-control algorithms suppress out-of-distribution Q-value overestimation.	`corl-team/CORL`	ReBRAC TD3-BC IQL	HalfCheetah-Medium-v2 Maze2D-Medium-v1 Walker2d-Medium-v2
Offline-to-Online Fine-Tuning Without Forgetting	Studies how offline-to-online reinforcement learning prevents forgetting and value collapse during continued interaction.	`corl-team/CORL`	IQL AWAC SPOT	Pen-Cloned-v1 Hammer-Cloned-v1 Hammer-Expert-v1
Off-Policy Actor-Critic for Continuous Control	Changes off-policy actor-critic update rules, losses, or exploration strategies to improve mean episodic return on continuous-control tasks.	`vwxyzjn/cleanrl`	DDPG TD3 SAC	HalfCheetah-v4 Reacher-v4 Ant-v4
On-Policy Actor-Critic for Continuous Control	Changes on-policy actor-critic objectives, update rules, or exploration mechanisms to improve mean episodic return on continuous-control tasks.	`vwxyzjn/cleanrl`	PPO AWR PPO (KL Penalty)	HalfCheetah-v4 Swimmer-v4 InvertedDoublePendulum-v4
Inverse RL Reward Learning from Demonstrations	Studies how reward models learned from expert demonstrations affect downstream policy return in continuous-control locomotion.	`HumanCompatibleAI/imitation`	GAIL AIRL BC	HalfCheetah-v4 Hopper-v4 Walker2d-v4
Value-Based Visual Control	Studies how value-based RL losses, update rules, and exploration strategies affect visual-control episodic return.	`vwxyzjn/cleanrl`	QR-DQN C51 Double-DQN	BreakoutNoFrameskip-v4 SeaquestNoFrameskip-v4 PongNoFrameskip-v4
Value-Based Discrete Control	Changes value estimation, uncertainty handling, or replay-based update rules to improve episodic return on discrete-action control tasks.	`vwxyzjn/cleanrl`	QR-DQN Dueling-DQN C51	CartPole-v1 LunarLander-v2 Acrobot-v1
Constraint Handling for Safe RL	Changes Lagrangian or controller-style multiplier updates and cost-reward advantage mixing to improve reward while keeping episode cost below target.	`PKU-Alignment/omnisafe`	Naive PPO Lagrangian PPO PID Lagrangian	SafetyPointGoal1-v0 SafetyCarGoal1-v0 SafetyPointButton1-v0
ML Systems & Efficient ML (SYS)				
Diffusion LM KV Cache Policy	Studies how token-state refresh intervals, masks, transfer ratios, and fallbacks affect denoising quality and cache reuse.	`maomaocun/dLLM-Cache`	Vanilla (Uncached) dLLM-Cache d2Cache Elastic-Cache	MATH-500 HumanEval ARC-Challenge
LLM KV Cache: Adaptive Quantization Policy	Studies adaptive 4-bit KV-cache quantization for instruction-tuned long-context inference, trading benchmark final-score quality against effective KV bits and compression.	`huggingface/transformers`	KIVI Overlap (4-bit) KVTuner-4 Per-Token KVTuner-4 KIVI SQuat Subspace (4-bit)	LongBench-E hotpotqa_e QA F1 LongBench-E passage_retrieval_en_e retrieval score LongBench-E repobench-p_e code-similarity score NeedleBench NIAH exact phrase retrieval GSM8K exact final-answer accuracy

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
LLM KV Cache Selection Budgeting	Studies how selection and eviction controllers allocate layer budgets and recent windows for quality, latency, and memory tradeoffs.	`huggingface/transformers`	Full Attention StreamingLLM Expected Attention LagKV	LongBench-E hotpotqa_e QA F1 LongBench-E passage_retrieval_en_e retrieval score LongBench-E repobench-p_e code-similarity score LongBench v2 train split multiple-choice accuracy GSM8K exact final-answer accuracy
LLM Pretraining: KV-Structural Reduction	Studies GPT-style KV-state structural reduction through MHA, MQA, GQA, and MLA-style latent KV compression under fixed nanoGPT pretraining.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	MHA MQA GQA MLA	ClimbMix val loss + KV bytes/token + WikiText-2/WikiText-103/LAMBADA heldout loss HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
LLM Pretraining: Custom GPU Kernel Optimization	Studies custom/fused MLP kernels for nanoGPT pretraining while preserving ClimbMix validation, held-out perplexity, and downstream lm-eval quality.	`karpathy/nanoGPT` `EleutherAI/lm-evaluation-harness`	ReLU-Squared (Torch) Triton GELU Triton ReLU-Squared (Fused)	ClimbMix val loss + WikiText-2/LAMBADA PPL HellaSwag, ARC-Easy, PIQA, WinoGrande 0-shot accuracy
LLM Post-Training Quantization (PTQ) Algorithm	Design a post-training quantization algorithm for a pretrained LLM that minimizes WikiText-2 perplexity degradation under INT4/INT3 group quantization without retraining.	`IST-DASLab/gptq`	Round-to-Nearest (RTN) GPTQ AWQ	PTQ INT4 PTQ INT3 PTQ INT4 (g64)
LLM Quantization-Aware Training (QAT) Algorithm	Design a quantization-aware training algorithm for a pretrained LLM that minimizes WikiText-2 perplexity after INT4/INT3/INT2 quantization at inference time.	custom	No QAT STE LSQ Finetune + PTQ	QAT INT4 QAT INT3 QAT INT2
Fused Attention Kernel Design for H100 GPUs	Design an OpenAI Triton fused self-attention forward kernel for H100 GPUs that maximizes throughput (TFLOPs/s) while preserving numerical correctness.	`Dao-AILab/flash-attention`	FlashAttention FlashAttention-2 FlashAttention-3	Head Dim 64 / Seq 4K Head Dim 128 / Seq 8K Head Dim 256 / Seq 16K
MoE Expert Parallelism Load Balancing	Design an efficient MoE expert-replica placement algorithm that minimizes GPU/node load imbalance while preserving inter-node locality and low runtime.	`deepseek-ai/eplb`	Greedy Zigzag Flat Zigzag	DeepSeek-V3 Qwen3-MoE DeepSeek-V2 Stress-Skew
Long-Context Inference-Time Sparse Attention	Design an inference-time sparse attention module for a pretrained instruction-tuned causal LLM that preserves NIAH and LongBench quality under a 25% density budget without retraining.	custom	Dense StreamingLLM BigBird Block Top-K	NIAH (8K) LongBench Qasper LongBench MultiFieldQA-EN
AI for Science (SCI)				
Mutation Fitness Predictor	Studies how mutant and wild-type protein representations can predict functional effects of sequence mutations.	`OATML-Markslab/ProteinGym`	Ridge Regression MLP Reshape CNN	BLAT_ECOLX ESTA_BACSU RASH_HUMAN

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Backbone-to-Sequence Inverse Folding	Studies how geometric structure encoding and sequence decoding recover amino-acid sequences from protein backbones.	`A4Bio/ProteinInvBench`	ProteinMPNN PiFold GVP	CATH 4.2 CATH 4.3 TS50
Geometric Protein Structure Encoder	Studies how local and global geometric protein representations transfer to structure-aware function prediction.	`a-r-j/ProteinWorkshop`	SchNet EGNN GearNet	EC GO-BP Fold
Atmospheric Column Emulator Architecture	Studies how neural emulator architecture maps vertical atmospheric states to sub-grid physics tendencies across training budgets.	`leap-stc/ClimSim`	CNN Encoder-Decoder U-Net HSR	Short Budget Medium Budget Long Budget
Diffusion-Prior Inverse Solver	Studies how diffusion priors and measurement guidance can be combined for inverse-problem reconstruction.	`devzhk/InverseBench`	DPS REDDiff LGD	Inverse Scattering Black Hole Imaging Inpainting
Molecular Representation Predictor	Studies how molecular graph and geometric representations improve property prediction under scaffold-based generalization.	`deepmodeling/Uni-Mol`	D-MPNN Uni-Mol GIN	BBBP BACE Tox21
Protein-Ligand Interaction Model	Studies how intra- and inter-molecular geometric interactions should be represented to predict binding affinity.	`guaguabujianle/EHIGN_PLA`	EHIGN GIGN SchNet EGNN	PDBbind 2013 PDBbind 2016 PDBbind 2019
Contrastive Virtual-Screening Objective	Studies how projection geometry and contrastive losses affect zero-shot protein-ligand screening quality.	`jianhuiwemi/HypSeek`	Vanilla CLIP HCC HCC + Hyperbolic Cone	HypSeek Training DUD-E LIT-PCBA DEKOIS 2.0
Weather Forecast Variable Aggregation	Studies how weather forecasting models aggregate information across heterogeneous meteorological variables for optimal prediction.	`microsoft/ClimaX`	Cross-Attention Mean Pooling Learned Weighted Sum	Z500 3-Day T850 5-Day 10m-Wind 7-Day
Industrial CFD Design: Custom Neural Operator Design	Designs and implements a custom neural operator for industrial aerodynamic design prediction on 3D unstructured point clouds.	`thuml/Neural-Solver-Library`	PointNet GraphSAGE Graph U-Net Transolver	Car Design AirfRANS Aircraft Design
Optimization & Theory (OPT)				
Optimization Bilevel	Studies a fixed bilevel-optimization benchmark based on Shen and Chen's penalty-based bilevel gradient descent experiments, selecting supported methods and tuning paper-style strategy hyperparameters.	`hanshen95/penalized-bilevel-gradient-descent`	V-PBGD G-PBGD RHG T-RHG	Toy Convergence HyperClean (Linear) HyperClean (MLP)
RAIN Convex-Concave	Studies gradient-norm convergence on the exact convex-concave benchmark instances used by the official RAIN bilinear and delta-function scripts.	`TrueNobility303/RAIN`	SEG R-SEG SEAG RAIN	Default Noise Low Noise High Noise
Optimizer Design for Diagonal-Net Sparse Recovery	Designs an optimizer that recovers a sparse linear predictor from fewer training samples under a diagonal-net parameterization with noisy labels.	`TrueNobility303/RAIN`	SGD AdaGrad Adam Adam (Alt.)	d=200, k=5, s=0.1 d=500, k=10, s=0.1 d=500, k=10, s=0.2 d=10000, k=50

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Differentially Private SGD: Privacy-Utility Optimization	Design an improved DP-SGD variant that achieves higher test accuracy under the same (epsilon, delta)-differential privacy budget.	custom	Standard DP-SGD Automatic Clipping (AUTO-S) Adaptive Quantile Clipping Step-Decay Noise Schedule	MNIST Fashion-MNIST CIFAR-10
Evolutionary Optimization Strategy Design	Design a novel combination of selection, crossover, mutation operators and/or evolutionary loop for continuous black-box optimization across multiple benchmark functions.	`DEAP/deap`	GA (SBX) CMA-ES Differential Evolution L-SHADE	Rastrigin (30D) Rosenbrock (30D) Ackley (30D) Rastrigin (100D)
Gradient Compression for Communication-Efficient Distributed Training	Design a gradient compression operator that reduces communication cost in distributed training while maintaining convergence quality.	custom	TopK Sparsification with Error Feedback QSGD (Quantized SGD) SignSGD	ResNet-20 / CIFAR-10 VGG-11-BN / CIFAR-100 ResNet-56 / CIFAR-10
Hyperparameter Optimization: Custom Search Strategy Design	Design a custom HPO strategy that improves final validation score and convergence under limited multi-fidelity evaluation budgets.	custom	Random Search TPE Hyperband DEHB BOHB Optuna CMA-ES	XGBoost SVM Neural Net
Multi-Objective Optimization: Custom Evolutionary Strategy Design	Design a custom multi-objective evolutionary strategy that improves convergence, diversity, and spread on standard benchmark problems.	`DEAP/deap`	NSGA-II MOEA/D SPEA2 NSGA-III RVEA AGE-MOEA	ZDT1 ZDT3 DTLZ2 DTLZ1
Sample-Efficient Neural Architecture Search	Design and implement a sample-efficient NAS optimizer that discovers high-performing architectures in the NAS-Bench-201 search space under a strict query budget.	`automl/naslib`	Random Search REA BANANAS	CIFAR-10 CIFAR-100 ImageNet16-120
Online Bandits: Exploration-Exploitation Strategy Design	Design and implement a bandit policy that minimizes cumulative regret across diverse multi-armed bandit settings.	`SMPyBandits/SMPyBandits`	UCB1 Thompson Sampling KL-UCB	Stochastic MAB Contextual Bandit Non-Stationary Bandit
PAC-Bayes Generalization Bound Optimization	Design a tighter PAC-Bayes generalization bound by optimizing the bound formulation, prior/posterior parameterization, and KL divergence estimation for stochastic neural networks.	`mperezortiz/PBB`	McAllester Catoni Quadratic	MNIST (FCN) MNIST (CNN) FashionMNIST (CNN)
Optimization Parity	Improve a fixed two-layer MLP's ability to learn sparse parity by designing only its initialization, training dataset, and AdamW hyperparameters.	`pytorch/examples`	Default Multi-Epoch No Weight Decay	n=32, k=8 n=50, k=8 n=64, k=8
Variance Reduction for Stochastic Optimization	Design an improved variance reduction strategy for stochastic gradient descent on finite-sum optimization problems.	custom	SVRG STORM STORM+	Logistic Regression MLP Ill-Conditioned
Classical & Adaptive Learning (CAL)				

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Few-Shot Image Classification Method	Studies how support encoding, query comparison, and loss design affect episodic few-shot image-classification accuracy.	`sicara/easy-few-shot-learning`	ProtoNet MatchingNet RelationNet	Mini-ImageNet 5w-5s CIFAR-FS CUB
Meta-Learning Inner-Loop Optimizer	Studies how differentiable inner-loop adaptation rules affect few-shot classification accuracy in gradient-based meta-learning.	`learnables/learn2learn`	MAML Meta-SGD ANIL	Mini-ImageNet 5w-1s Mini-ImageNet 5w-5s CIFAR-FS 5w-5s
Pool-Based Active Learning Query Strategy	Studies how unlabeled-sample query rules affect accuracy under a fixed labeling budget.	`JordanAsh/badge`	BADGE BAIT BALD Least Confidence Random	Letter Spambase Splice
Unsupervised Tabular Anomaly Detector	Studies how unlabeled anomaly scoring algorithms identify outliers across tabular data distributions.	custom	IF (Isolation Forest) LOF OCSVM ECOD COPOD	Cardio Thyroid Satellite Shuttle
Post-Hoc Probability Calibration Mapping	Studies how post-hoc probability transforms improve classifier confidence calibration.	custom	Platt Temperature Scaling Isotonic Regression	RF / MNIST MLP / Fashion-MNIST GBM / Madelon SVM / Breast Cancer
Geometry-Robust Clustering Algorithm	Studies how clustering objectives and distance metrics handle convex blobs, non-convex moons, and high-dimensional digit data.	custom	K-Means DBSCAN HDBSCAN	Blobs Moons Digits
Continual Learning Importance Regularizer	Changes parameter-importance estimation and regularization loss to reduce catastrophic forgetting and improve final average accuracy across contexts.	`GMvandeVen/continual-learning`	EWC SI Online EWC	Split-MNIST Permuted-MNIST Split-CIFAR100
Nonlinear 2D Structure-Preserving Embedding	Studies how nonlinear dimensionality reduction preserves neighborhood structure in low-dimensional embeddings.	custom	PCA t-SNE UMAP TriMap PaCMAP	MNIST Fashion-MNIST 20 Newsgroups
Adaptive Boosting Weight and Target Strategy	Studies how pseudo-targets, learner weights, and sample reweighting affect boosted ensemble performance.	custom	AdaBoost Gradient Boosting XGBoost-style	Breast Cancer Diabetes California Housing
Heterogeneous Federated Server Aggregation	Changes server-side client selection and model aggregation to improve federated test accuracy under heterogeneous client data.	`adap/flower`	FedAvg FedProx SCAFFOLD	CIFAR-10 (Non-IID alpha=0.1) FEMNIST Shakespeare
Correlation-Aware Tabular Imputation	Studies how feature correlations and predictive structure guide missing-value imputation in tabular data.	custom	Mean Imputation KNN Imputation MICE MissForest GAIN	Breast Cancer Wisconsin Wine California Housing
Selective Deferral Under Subgroup Shift	Studies how acceptance and deferral rules trade off selective risk, subgroup robustness, and coverage on AIF360 tabular datasets.	custom	Confidence Thresholding Conformal Abstention Learned Deferral Group-wise Thresholding	Adult COMPAS Law School GPA

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Shift-Robust Subgroup Calibration	Studies how post-hoc calibration behaves under subgroup distribution shift and worst-group reliability constraints on AIF360 tabular datasets.	custom	Temperature Scaling Isotonic Regression Beta Calibration Group-wise Temperature Scaling	Adult COMPAS Law School GPA
Genetic Programming Search for Symbolic Regression	Studies how symbolic-regression search strategies recover generalizable analytical expressions.	`trevorstephens/gplearn`	Standard GP Parsimony GP Lexicase GP	Nguyen-7 Nguyen-10 Koza-3
Deep Learning (DL)				
Adaptive Classification Loss	Modify the training loss over logits and labels to improve classification accuracy across image-model families.	custom	Label Smoothing Focal Loss PolyLoss	ResNet-56 / CIFAR-100 VGG-16-BN / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Image Augmentation Policy	Design the training transform pipeline combining geometric, photometric, and erasing operations to improve image-classification generalization.	custom	Cutout RandAugment TrivialAugmentWide	ResNet-20 / CIFAR-10 ResNet-56 / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Hierarchical Classification Loss Weighting	Studies how fine-label and coarse-label objectives should be combined to improve hierarchical image classification.	custom	Uncertainty Weighting DWA PCGrad	ResNet-20 / CIFAR-100-MT ResNet-56 / CIFAR-100-MT VGG-16-BN / CIFAR-100-MT
Spatial Feature Aggregation	Studies how global spatial features should be aggregated to improve image-classification accuracy across convolutional architectures.	custom	Global Max GeM Avg + Max	ResNet-56 / CIFAR-100 VGG-16-BN / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Long-Tail Class Reweighting	Studies how class-count statistics should be mapped to loss weights to improve test accuracy on balanced test sets for long-tailed image classification.	custom	Inverse Frequency Class-Balanced (Effective Number) Balanced Softmax	ResNet-32 / CIFAR-10-LT ResNet-32 / CIFAR-100-LT VGG-16-BN / CIFAR-100-LT
Convolutional Activation Nonlinearity	Studies how drop-in activation functions affect accuracy across convolutional image classifiers.	custom	GELU SiLU Mish	ResNet-20 / CIFAR-10 VGG-16-BN / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Architecture-Aware Learning-Rate Scheduling	Designs an epoch-level learning-rate curve conditioned on architecture and dataset to improve convergence and final classification accuracy.	custom	Cosine WarmupCosine OneCycle	ResNet-20 / CIFAR-10 ResNet-56 / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Normalization Statistics and Affine Design	Studies how normalization statistics and affine behavior affect convolutional training stability and test accuracy.	custom	GroupNorm Batch-Instance Norm Switchable Norm	ResNet-56 / CIFAR-100 ResNet-110 / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Adaptive Regularization Loss	Adds a model-, output-, input-, or epoch-dependent regularization term to improve classification generalization beyond standard weight decay.	custom	DropBlock Confidence Penalty Orthogonal Regularization	ResNet-56 / CIFAR-100 VGG-16-BN / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Residual Block Skip Design	Studies how shortcut transformations and residual branch computation affect optimization and generalization across network depths.	custom	Pre-Activation Gated Residual Stochastic Depth	ResNet-20 / CIFAR-10 ResNet-56 / CIFAR-100 ResNet-110 / CIFAR-100

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
DL Weight Initialization Strategy Design	Designs data-independent initialization for convolutional, normalization, and classifier layers to improve convergence and final accuracy.	custom	Kaiming Normal Fixup Orthogonal	ResNet-56 / CIFAR-100 VGG-16-BN / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Time Series & Forecasting (TS)				
Concept-Drift-Aware Quantitative Forecasting	The stock prediction model and data pipeline are redesigned to handle temporal distribution shift and improve signal quality and portfolio metrics.	`microsoft/qlib`	TRA AdaRNN LightGBM	CSI 300 CSI 300 (Shifted) CSI 300 (Recent)
Graph-Based Quantitative Forecasting	Studies how inter-asset graph relationships affect return signal quality and portfolio performance.	`microsoft/qlib`	HIST GATs LightGBM	CSI 300 CSI 100 CSI 300 (Recent)
Quantitative Return Forecasting	Studies how predictive models and input processing affect next-period return signals and portfolio performance.	`microsoft/qlib`	LightGBM LSTM Transformer	CSI 300 CSI 100 CSI 300 (Recent)
Spatial-Temporal Traffic Forecasting Model	Studies how spatial-temporal models capture sensor-network dependencies for traffic forecasting.	`GestaltCogTeam/BasicTS`	STID DLinear StemGNN iTransformer TimesNet SOFTS TimeMixer	METR-LA PEMS-BAY PEMS04
Reconstruction Model for Time-Series Anomaly Detection	An unsupervised reconstruction model detects anomalous multivariate time-series segments to improve F-score.	`thuml/Time-Series-Library`	DLinear TimesNet PatchTST	PSM MSL SMAP
Multivariate Time-Series Classification Model	Studies how representation learning improves classification of multivariate time-series signals.	`thuml/Time-Series-Library`	DLinear TimesNet PatchTST	EthanolConcentration FaceDetection Handwriting
Exogenous-Variable Target Forecasting Model	Studies how exogenous variables improve target-channel forecasting.	`thuml/Time-Series-Library`	DLinear PatchTST iTransformer TimeXer	ETTh1 Weather ECL
Masked Multivariate Time-Series Imputation	Studies how imputation models reconstruct missing regions in multivariate time series.	`thuml/Time-Series-Library`	DLinear TimesNet PatchTST	ETTh1 (25% missing) Weather (25% missing) ECL (25% missing)
Multivariate Long-Horizon Forecasting Model	Studies how long-horizon forecasting models predict future multivariate sequences.	`thuml/Time-Series-Library`	DLinear PatchTST iTransformer TimeMixer TimeXer	ETTh1 Weather ECL
Univariate Short-Horizon Forecasting Model	Studies how short-horizon forecasting models predict seasonal univariate series.	`thuml/Time-Series-Library`	DLinear TimesNet PatchTST TimeMixer	M4 Monthly M4 Quarterly M4 Yearly
Structured & Causal Reasoning (SCR)				

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Discrete Causal Graph Discovery	Studies how causal discovery algorithms recover equivalence-class graph structure from discrete observational data.	`py-why/causal-learn`	PC GES GRaSP BOSS Hill Climbing	Cancer Child ALARM HAILFINDER Win95pts
Linear Gaussian Causal Discovery	Studies how observational algorithms recover causal graph structure under linear Gaussian assumptions.	`py-why/causal-learn`	PC GRaSP BOSS	ER (n=10) ER (n=20) SF (n=50) SF (n=50, Hard) ER (n=20, Noisy)
Non-Gaussian Causal Discovery	Studies how non-Gaussian structure can identify directed causal relationships from observational data.	`py-why/causal-learn`	ICA-LiNGAM DirectLiNGAM NOTEARS	ER (n=30) ER (n=50) SF (n=100)
Nonlinear Causal Discovery	Studies how nonlinear additive-noise assumptions support directed causal graph recovery from observations.	`py-why/causal-learn`	CAM NOTEARS-MLP DirectLiNGAM GraN-DAG	SF (n=20, GP) ER (n=20, Gauss) ER (n=12, Low-Sample)
Heterogeneous Treatment Effect Estimation	Studies how observational estimators recover individual and average treatment effects on synthetic CATE benchmark families.	custom	S-Learner T-Learner IPW Causal Forest DR-Learner R-Learner	IHDP-inspired Synth Jobs/LaLonde-inspired Synth ACIC-inspired Synth
Unconditional Graph Generator Architecture	Studies how graph generator architecture affects distributional match to target graph statistics.	`pyg-team/pytorch_geometric`	GraphVAE GRAN DiGress	Community-Small Ego-Small ENZYMES
Structure-Aware Graph Readout Pooling	Studies how graph-level readout mechanisms affect graph classification accuracy and macro F1 under a fixed message-passing backbone.	`pyg-team/pytorch_geometric`	GIN + Sum SAGPool DiffPool	MUTAG PROTEINS NCI1
Graph Link Encoder-Decoder	Studies how node encoders and edge decoders affect missing-link prediction quality.	custom	GCN + MLP Decoder VGAE SEAL	Cora CiteSeer ogbl-collab
Graph Node Message Passing	Studies how message-passing layers affect node classification across citation network benchmarks.	`pyg-team/pytorch_geometric`	GCN GAT GraphSAGE	Cora CiteSeer PubMed
Homophily-Heterophily Graph Filter	The graph signal propagation filter is changed to improve node classification accuracy across homophilic and heterophilic graphs.	`ivam-he/ChebNetII`	GPR-GNN BernNet ChebNetII	Cora CiteSeer Texas Cornell
Trustworthy Learning (TL)				
Score-Based Black-Box Linf Attack	Designs a query-efficient black-box Linf evasion attack to improve attack success rate under a fixed per-sample query budget.	`Harry24k/adversarial-attacks-pytorch`	Square Attack SPSA Random Search	ResNet-20 / CIFAR-10 VGG-11-BN / CIFAR-10 MobileNet-V2 / CIFAR-10 ResNet-20 / CIFAR-100 MobileNet-V2 / CIFAR-100
Sparse L0 Adversarial Attack	Studies how sparse perturbation strategies improve attack success while respecting a strict pixel budget.	`Harry24k/adversarial-attacks-pytorch`	OnePixel SparseFool JSMA Pixle Sparse-RS	ResNet-20 / CIFAR-10 VGG-11-BN / CIFAR-10 MobileNet-V2 / CIFAR-10 ResNet-20 / CIFAR-100 MobileNet-V2 / CIFAR-100
White-Box Linf Evasion Attack	Designs a gradient-based white-box Linf attack to improve attack success rate while respecting the perturbation budget.	`Harry24k/adversarial-attacks-pytorch`	FGSM PGD MI-FGSM AutoAttack	ResNet-20 / CIFAR-10 VGG-11-BN / CIFAR-10 ResNet-20 / CIFAR-100 VGG-11-BN / CIFAR-100 MobileNet-V2 / CIFAR-100

(continued on next page)

(Table 4 continued from previous page)

Name	Description	External Package(s)	Baselines	Evaluation Settings
Linf Adversarial Training for Robust Accuracy	Studies how adversarial training procedures improve robust accuracy while maintaining clean accuracy.	`Harry24k/adversarial-attacks-pytorch`	Standard Training PGD-AT TRADES MART AWP + TRADES	SmallCNN / MNIST PreAct ResNet-18 / CIFAR-10 VGG-11-BN / CIFAR-10 PreAct ResNet-18 / CIFAR-100
Poisoned-Sample Scoring for Backdoor Filtering	A suspicion scoring rule identifies and filters backdoored training examples to reduce attack success rate while preserving clean accuracy.	custom	Confidence Filter Spectral Signatures Activation Clustering Z-Score Outlier	ResNet-20 / CIFAR-10 (BadNets) VGG-16-BN / CIFAR-100 (Blend) MobileNet-V2 / Fashion-MNIST (BadNets)
Targeted Update Rules for Class Unlearning	An unlearning update rule removes forget-class information while improving retained accuracy and reducing forget-set membership leakage.	custom	Retain Fine-Tune Negative Gradient Bad Teacher SCRUB	ResNet-20 / CIFAR-10 (Class 0) VGG-16-BN / CIFAR-100 (Class 0) MobileNet-V2 / Fashion-MNIST (Class 0)
Training Regularization for Membership Privacy	Studies how privacy-preserving training losses reduce membership leakage while maintaining accuracy.	custom	ERM Label Smoothing Confidence Penalty RelaxLoss	ResNet-20 / CIFAR-10 VGG-16-BN / CIFAR-100 MobileNet-V2 / Fashion-MNIST
Robust Losses for Label-Flip Poisoning	A robust loss or sample-weighting rule improves clean accuracy under label-flip poisoning and reduces poisoned-label memorization.	custom	Cross-Entropy Generalized Cross-Entropy Symmetric Cross-Entropy Bootstrap	ResNet-20 / CIFAR-10 (Label-Flip) VGG-16-BN / CIFAR-100 (Label-Flip) MobileNet-V2 / Fashion-MNIST (Label-Flip)

B. MLS-Bench-Lite: 30-Task Subset

The 30 tasks of MLS-Bench-Lite, grouped by the 12 MLS-Bench domains, are:

AI for Science. Mutation Fitness Predictor; Diffusion-Prior Inverse Solver; Protein-Ligand Interaction Model.

Structured & Causal Reasoning. Discrete Causal Graph Discovery; Unconditional Graph Generator Architecture.

Vision & Generation. 3D Scene Densification Strategy; Low-Step Diffusion Bridge Sampling; Frequency-Aware Autoencoding Loss.

Deep Learning. Spatial Feature Aggregation; Convolutional Activation Nonlinearity.

Robotics. Latent World-Model Planner; Guided Diffusion Sampling for Robot Actions; Diffusion Policy Learning for Robot Control; Humanoid Transfer Policy Learning; Behavioral Cloning Loss for Manipulation.

Language Models. Masked Diffusion Demasking Policy; Pretraining Optimizer Design; Reasoning RL Importance-Sampling Granularity.

ML Systems & Efficient ML. Post-Training Weight Quantization; Quantization-Aware Language-Model Training; Long-Context Inference-Time Sparse Attention.

Classical & Adaptive Learning. Geometry-Robust Clustering Algorithm; Nonlinear 2D Structure-Preserving Embedding.

Optimization & Theory. Multi-Objective Evolutionary Survival and Variation; Variance-Reduced Stochastic Optimization.

Time Series & Forecasting. Concept-Drift-Aware Quantitative Forecasting; Exogenous-Variable Target Forecasting Model; Masked Multivariate Time-Series Imputation.

Reinforcement Learning. Value-Based Discrete Control.

Trustworthy Learning. Training Regularization for Membership Privacy.

C. Agent Prompts and Tool Schemas

C.1. System Prompt

The default scientific-innovation system prompt used in our main agent runs is reproduced below.

```
You are an ML scientist. Your goal is to propose and implement a novel algorithmic contribution that improves performance on the given
      task.

What counts as a good contribution:
- A new loss function or objective formulation
- A new policy update rule or gradient estimation method
- A novel exploration or regularization strategy
- A new way to parameterize or combine components, with clear motivation

What does NOT count:
- Trivially increasing network capacity to brute-force a metric (a per-task parameter cap is enforced before each test)
- Hyperparameter tuning (learning rates, batch sizes, etc.)
- Copying a reference baseline with cosmetic changes
- Pure engineering tricks without algorithmic novelty

Parameter count is capped (enforced before each test); architectural changes within that budget are encouraged.

IMPORTANT workflow:
1. FIRST call edit() to implement your improved algorithm. Do NOT call test() before making edits.
2. THEN call test() to run the experiment. Each run is numbered (#1, #2, ...).
3. Review the metrics, then edit() to improve your solution based on the feedback.
4. Call test() again to verify the improvement. You MUST iterate at least once (edit → test → review → edit → test) before
      submitting, unless only 1 test is allowed.
5. When satisfied, call submit(n=N) to submit your best test #N as final.
You have a limited number of test() calls, so make each one count by editing first.

Available tools:
- edit(op, filename, content, ...): Modify files in the workspace.
  - op='replace': replace lines start_line..end_line with content
  - op='insert': insert content after after_line
  - op='create': create a new file (only if allow_create=true)
- test(): Run a new experiment. Executes training and evaluation. Each run is
  numbered #1, #2, etc. The first test runs all seeds; intermediate tests run one seed.
  You have a limited budget of test() calls, so make each one count by editing first.
  If max tests is reached, the last test is auto-submitted.
- submit(n=N): Submit the result from test #N as your final answer (1-indexed).
  This does NOT re-run anything — it picks a previous result. Use n=-1 for the latest.
  You must call test() at least once before you can submit.
- undo(n=1): Revert the last n edit operations.

Constraints:
- Each file shown in the prompt is labeled [READ-ONLY] or [EDITABLE — lines X-Y only].
  Only edit files and line ranges marked EDITABLE. Do not touch READ-ONLY files.
- When a file has multiple editable regions, editing one region may shift line numbers in subsequent regions. Edit from the last (bottom-
      most) region first, or check the updated editable ranges shown after each edit.
- You MUST call test() at least once before you can call submit().
- When you are done, call submit(n=N) to submit your best test result.
- If your algorithm requires new hyperparameters (e.g., cql_alpha, expectile_tau) that are not
  in the existing config, hardcode their values directly in your code (e.g., in __init__).
  You cannot modify the training loop or config to pass them via command line.
```

The no-budget ablation replaces the novelty and parameter-budget preamble with the shorter `SYSTEM_PROMPT_SCI_PREAMBLE_NOBUDGET` variant.

C.2. Initial User Prompt

The initial user prompt is assembled from task metadata, annotated files, evaluation commands, baseline results, and budget information as follows.

```
[Optional extra context block]
# <BASELINE_DERIVATIONS_OR_DEEP_THEORETICAL_CONTEXT> (reference material)

<EXTRA_CONTEXT_TEXT>

# Task: <TASK_NAME>

<TASK_DESCRIPTION>

## <FILENAME>  [READ-ONLY — do not edit]
```<LANGUAGE>
```
```

```
<LINE_NUMBERED_READ_RANGE_OR_FULL_FILE_CONTENT>
```

<FILENAME> [EDITABLE — lines <START>–<END> only]
```<LANGUAGE>
Lines <START>-<END>:
<LINE_NUMBERED_READ_RANGE>
```

[If allow_hack=true, editability annotations are omitted. If rigorous_codebase=true and baseline variants are available, read-only non-editable files are skipped.]

<BASELINE_NAME> baseline — editable region [READ-ONLY — reference implementation]
```python
Lines <START>–<END>:
<LINE_NUMBERED_BASELINE_EDITABLE_REGION_WITH_THREE_LINES_OF_CONTEXT>
```

Evaluation Commands
Your algorithm is evaluated by running:
  - `<COMMAND>` → label: `<LABEL>`

Compute Budget
All evaluation runs on **NVIDIA H100 80GB** GPU(s). Your algorithm must complete within the time limits below. If a command exceeds its time limit, the run is killed and the result is **invalid** (it will not count as a valid test result). Design your model to be efficient enough to train and evaluate within these constraints.

Command	GPUs	Time Limit
`<LABEL>`	<GPU_DESCRIPTION>	<TIME_LIMIT>

Baseline Results
Beat these with your algorithm:

baseline	<LABEL_1>	<LABEL_2>
<BASELINE_NAME>	<METRIC_VALUE>	<METRIC_VALUE>

Your Budget
- **Action budget**: <MAX_STEPS> total tool calls (every edit / test / undo / web_search / web_extract counts; submit does not)
- **Test invocations**: at most <MAX_TESTS> (each test() call also consumes one action from the budget above)
- You **must** iterate at least once (edit → test → review → edit → test) before submitting.

[If <MAX_TESTS>=1, the iteration line is replaced by:]
- **CRITICAL — single-test mode (max_tests=1)**: your one and only `test()` call is automatically the FINAL submission. Whatever metrics it returns are written to the leaderboard; whatever bugs it hits are recorded as a failed submission with empty metrics — there is **no second chance**.
- Before you call `test()`, run through this checklist mentally: (1) tensor shapes match between your model layers and the expected input/output, (2) dtypes / device are consistent, (3) any new module is actually used in `forward()`, (4) loss is finite on a tiny dummy input, (5) you handled the corner cases the task description warns about.
- If you are unsure, spend remaining edit budget tightening the code rather than rushing to test. **A crashed test is a wasted submission.** Only call `test()` when you can defend each line.

C.3. Tool Schemas

The core workspace tool schemas below are reproduced for the three tools `edit`, `test`, and undo.

```
{
  "name": "edit",
  "description": "Edit files in the workspace. Three operations are supported:\n  create: Create a new file with the given content. Only available if allow_create=true.\n  insert: Insert one or more lines immediately after `after_line` (1-indexed).\n  replace: Replace lines `start_line`..`end_line` (inclusive, 1-indexed) with `content`.\nFile paths are relative to the package root (e.g. 'LLaMA-Factory/src/...').\nLines within protected ranges must NOT be modified.",
  "input_schema": {
    "type": "object",
    "properties": {
      "op": {
        "type": "string",
        "enum": [
          "create",
          "insert",
          "replace"
        ],
        "description": "The edit operation to perform."
      },
      "filename": {
        "type": "string",
        "description": "Package-relative path to the file (e.g. 'LLaMA-Factory/src/llamafactory/train/dpo/trainer.py')."
```

```
      },
      "content": {
        "type": "string",
        "description": "Content to write (for create/replace) or insert."
      },
      "after_line": {
        "type": "integer",
        "description": "Line number after which to insert (required for op='insert')."
      },
      "start_line": {
        "type": "integer",
        "description": "First line to replace, 1-indexed (required for op='replace')."
      },
      "end_line": {
        "type": "integer",
        "description": "Last line to replace, 1-indexed inclusive (required for op='replace')."
      }
    },
    "required": [
      "op",
      "filename",
      "content"
    ]
  }
}
```

```
{
  "name": "test",
  "description": "Run a new experiment. Executes training and evaluation, then returns metrics. Each run is numbered #1, #2, etc. All
      runs use all configured seeds. You have a limited test budget.",
  "input_schema": {
    "type": "object",
    "properties": {}
  }
}
```

```
{
  "name": "undo",
  "description": "Revert the last n file modification actions (create/insert/replace) by restoring pre-edit snapshots. Does not undo test
       calls.",
  "input_schema": {
    "type": "object",
    "properties": {
      "n": {
        "type": "integer",
        "description": "Number of edit actions to undo (default: 1)."
      }
    }
  }
}
```

D. Test-Time Scaling Configurations

Sampling and exploration. The two ReAct-based setups reuse the default scaffold and per-call sampling defaults of the underlying provider; we only vary the action budget. *Sampling* runs 16 independent ReAct chains of at most 5 actions, each ending in a single `test` call, and reports the running best across them. *Exploration* runs a single 50-action ReAct chain that may issue up to 16 `test` calls, iteratively refining one solution. All runs use $\text{seed} = 42$ and a single in-process container runtime.

OpenEvolve (test-time evolution)

OpenEvolve is run with a 160-LLM-call budget, two calls per iteration (mutation followed by a judge that contributes a feedback weight to the score). Table 5 lists the full hyperparameter set.

Table 5. OpenEvolve hyperparameters used for the test-time-evolution setup.

Parameter	Value
Budget	
LLM call budget	160
Max iterations	80
Calls per iteration	2 (mutation + judge)
LLM feedback (judge)	enabled
LLM sampling	
Temperature	0.8
Top-p	0.95
Max tokens	16 000
LLM timeout (s)	240
Prompt construction	
Top programs in prompt	3
Diverse programs in prompt	2
Include execution artifacts	yes
Max artifact bytes	20 480
Meta-prompting	enabled
Meta-prompt weight	0.15
Database / population	
Population size	60
Archive size	30
Number of islands	2
Migration interval (iters)	15
Migration rate	0.15
Elite selection ratio	0.20
Exploration ratio	0.30
Exploitation ratio	0.70
Diversity metric	feature-based
Feature dimensions	score, complexity
Feature bins	10
Evaluator	
Evaluation timeout (s)	3 600
Parallel evaluations	1
LLM feedback weight	0.20
Enable artifacts	yes

TTT-Discover (test-time training)

TTT-Discover fine-tunes the underlying policy with LoRA-based RL using each task's evaluator as the reward signal. It is run on `Qwen3.5-35B-A3B` (MoE).

Table 6. TTT-Discover hyperparameters used in the test-time-training setup.

Parameter	**Value**
Model and runtime	
Base model	`Qwen3.5-35B-A3B` (MoE)
Container runtime	local (in-process)
Compute scale	0.5
Seeds	{42}
Reasoning effort	high (thinking enabled)
Thinking-token budget	10 000
RL training loop	
Number of epochs	50
Group size (rollouts per group)	4
Groups per batch	1
Phase-1 max tokens	64 000
KL penalty coefficient	0.30
Learning rate	5×10^{-6}
LoRA rank	32
Edit-format penalty	0.0
Checkpoint interval	every 2 epochs

E. Task Subsets for Ablation and Analysis Experiments

The ablation and analysis experiments in Sections 4.3 and 5 are each evaluated on a curated subset, chosen so that the relevant property is well defined on every included task.

Scientific innovation vs. engineering optimization (Sec. 4.3, Figure 5 left). Four tasks where the same editable region admits both a scientific-innovation prompt and an engineering-optimization prompt:

- Quantitative Return Forecasting
- Graph Node Message Passing
- Multivariate Long-Horizon Forecasting Model
- Residual Block Skip Design

Capacity-budget validity control (Sec. 4.3, Figure 5 middle). Four computer-vision and reinforcement-learning tasks where the agent can change model size and the budget check is therefore meaningful:

- Value-Based Discrete Control
- Off-Policy Actor-Critic for Continuous Control
- Diffusion Model Architecture Design
- Class-Conditional Diffusion: Conditioning Injection Methods

Test-time scaling (Sec. 5.1). Six low-latency tasks for the inference-only setups (Scaling Sampling, Scaling Exploration, OpenEvolve):

- Optimization Bilevel
- Variance Reduction for Stochastic Optimization
- Long-Tail Class Reweighting
- Homophily-Heterophily Graph Filter
- Reconstruction Model for Time-Series Anomaly Detection
- Heterogeneous Treatment Effect Estimation

The TTT-Discover test-time-training setup uses the first two for training.

Verifier-limited compute allocation (Sec. 5.2). Five LLM pretraining tasks:

- Transformer Feed-Forward Block
- Pretraining Learning-Rate Schedule
- Subquadratic Attention Mechanism
- Pretraining Optimizer Design
- Normalization and Block Layout

Context engineering (Sec. 5.3). Nine tasks across optimization, graph, RL, computer vision, and AI-for-science.

- Homophily-Heterophily Graph Filter
- PAC-Bayes Generalization Bound Optimization
- Variance Reduction for Stochastic Optimization

- Genetic Programming Search for Symbolic Regression
- Diffusion-Prior Inverse Solver
- Diffusion Model: Sampler Efficiency Optimization
- Diffusion Prediction Parameterization
- Intrinsic Exploration for Sparse Rewards
- Q-Overestimation Suppression for Offline Continuous Control

F. Human Expert Assessment

This appendix accompanies the case-study in Section 5 with per-task examples. Each task block first restates the research question, then shows the editable region of the template the agent starts from, then shows one strong human baseline , and finally one or more curated agent submissions. Each agent submission is preceded by the model name and a brief description and followed by the human evaluator's written assessment. To keep the appendix readable, every listing is restricted to the editable region or a focused excerpt, since full agent submissions can run to several hundred lines.

The colored bars in the left margin of every code listing mark how each line relates to the editable region of the original template: green — the line was modified by the agent or baseline shown in this listing; blue — the line is inside the editable region but unchanged from the template; no bar — the line is outside the editable region (read-only context shown for orientation).

F.1. Fused Causal Attention Kernel

Task. A fused causal self-attention forward pass in OpenAI Triton is evaluated on NVIDIA H100, maximizing throughput (TFLOPs/s) while keeping the maximum absolute error below 10^{-2} against a reference. Editable: the Triton kernel and its Python wrapper. Read-only: the benchmark harness, FLOP accounting, and correctness check. Provided baselines include a naive Triton kernel, a Flash-Attention v2 style two-pass causal kernel, and a Flash-Attention v3 reference.

```
29 @triton.jit
30 def _custom_attn_fwd(
31     Q, K, V, Out,
32     sm_scale,
33     stride_qh, stride_qm, stride_qk,
34     stride_kh, stride_kn, stride_kk,
35     stride_vh, stride_vn, stride_vk,
36     stride_oh, stride_om, stride_ok,
37     seqlen,
38     BLOCK_M: tl.constexpr,
39     BLOCK_N: tl.constexpr,
40     BLOCK_DMODEL: tl.constexpr,
 ... 66 lines elided ...
107     grid = (triton.cdiv(seqlen, BLOCK_M), batch * nheads)
108     _custom_attn_fwd[grid](
109         q, k, v, o, sm_scale,
110         q.stride(1), q.stride(2), q.stride(3),
111         k.stride(1), k.stride(2), k.stride(3),
112         v.stride(1), v.stride(2), v.stride(3),
113         o.stride(1), o.stride(2), o.stride(3),
114         seqlen,
115         BLOCK_M=BLOCK_M, BLOCK_N=BLOCK_N,
116         BLOCK_DMODEL=headdim, IS_CAUSAL=causal,
117     )
118     return o
119
```

```
27 # ================================================================
28
 - editable region begins at line 29 -
29 @triton.autotune(
30     configs=[
31         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
32         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
33         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=4, num_warps=8),
34         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=3, num_warps=4),
35         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=4, num_warps=8),
36         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
37         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 32}, num_stages=3, num_warps=4),
38         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 32}, num_stages=4, num_warps=4),
39     ],
40     key=['seqlen', 'BLOCK_DMODEL', 'IS_CAUSAL'],
41 )
```

```
@triton.jit
def _flash_v3_fwd(
    Q, K, V, Out,
    stride_qh, stride_qm, stride_qk,
    stride_kh, stride_kn, stride_kk,
 ... 5 lines elided ...
    BLOCK_DMODEL: tl.constexpr,
    IS_CAUSAL: tl.constexpr,
):
    """FA3-inspired: autotuned two-pass causal with software pipelining."""
    start_m = tl.program_id(0)
    off_hz = tl.program_id(1)

 ... 6 lines elided ...
    offs_n = tl.arange(0, BLOCK_N)
    offs_d = tl.arange(0, BLOCK_DMODEL)

    # Load Q with scale already fused (done in wrapper)
    q_ptrs = Q + q_offset + offs_m[:, None] * stride_qm + offs_d[None, :] * stride_qk
 ... 39 lines elided ...
        k = tl.load(k_ptrs, mask=(start_n + offs_n[:, None]) < seqlen, other=0.0)
        qk = tl.dot(q, tl.trans(k))
        qk = tl.where(offs_m[:, None] >= (start_n + offs_n[None, :]), qk, float("-inf"))
        m_ij = tl.max(qk, axis=1)
        m_new = tl.maximum(m_i, m_ij)
        alpha = tl.math.exp2(m_i - m_new)
        p = tl.math.exp2(qk - m_new[:, None])
        l_i = l_i * alpha + tl.sum(p, axis=1)
        acc = acc * alpha[:, None]
        v_ptrs = V + v_offset + (start_n + offs_n[:, None]) * stride_vn + offs_d[None, :] * stride_vk
        v = tl.load(v_ptrs, mask=(start_n + offs_n[:, None]) < seqlen, other=0.0)
        acc += tl.dot(p.to(v.dtype), v)
        m_i = m_new

    acc = acc / l_i[:, None]
    o_ptrs = Out + o_offset + offs_m[:, None] * stride_om + offs_d[None, :] * stride_ok
    tl.store(o_ptrs, acc.to(Out.dtype.element_ty), mask=offs_m[:, None] < seqlen)


def custom_attention_forward(q, k, v, causal=True, sm_scale=None):
    """FA3-inspired: autotuned pipelining + fused scale + two-pass causal."""
    batch, nheads, seqlen, headdim = q.shape
    q, k, v = q.contiguous(), k.contiguous(), v.contiguous()
    if sm_scale is None:
        sm_scale = 1.0 / math.sqrt(headdim)
    # Fuse scale into Q
    q = (q * (sm_scale * 1.44269504)).contiguous()
    o = torch.empty_like(q)
    grid = lambda META: (triton.cdiv(seqlen, META['BLOCK_M']), batch * nheads)
    _flash_v3_fwd[grid](
        q, k, v, o,
        q.stride(1), q.stride(2), q.stride(3),
        k.stride(1), k.stride(2), k.stride(3),
        v.stride(1), v.stride(2), v.stride(3),
```

Agent: Claude Opus 4.6.

Causal/non-causal split with H100-tuned autotune configs.

```
# ============================================================

 - editable region begins at line 29 -
@triton.autotune(
    configs=[
        # Large BLOCK_M configs (good for small headdim like 64)
        triton.Config({'BLOCK_M': 256, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
        triton.Config({'BLOCK_M': 256, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=4, num_warps=8),
```

```
        # Medium configs (good for headdim 128)
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=4, num_warps=8),
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=3, num_warps=4),
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
        # Smaller configs (good for large headdim like 256)
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=3, num_warps=4),
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=4, num_warps=8),
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 32}, num_stages=3, num_warps=4),
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 32}, num_stages=4, num_warps=4),
        # Additional hdim256 configs - split work more to reduce register pressure
        triton.Config({'BLOCK_M': 32, 'BLOCK_N': 64}, num_stages=3, num_warps=4),
        triton.Config({'BLOCK_M': 32, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
        triton.Config({'BLOCK_M': 16, 'BLOCK_N': 128}, num_stages=3, num_warps=4),
        triton.Config({'BLOCK_M': 16, 'BLOCK_N': 64}, num_stages=3, num_warps=4),
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=2, num_warps=4),
        triton.Config({'BLOCK_M': 64, 'BLOCK_N': 128}, num_stages=2, num_warps=8),
        triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=2, num_warps=8),
    ],
    key=['seqlen', 'BLOCK_DMODEL', 'IS_CAUSAL'],
)
@triton.jit
def _custom_attn_fwd(
    Q, K, V, Out,
 ... 61 lines elided ...
    else:
        hi = non_causal_end

    for start_n in range(non_causal_end, hi, BLOCK_N):
        start_n = tl.multiple_of(start_n, BLOCK_N)
        k_ptrs = K + k_offset + (start_n + offs_n[:, None]) * stride_kn + offs_d[None, :] * stride_kk
        k = tl.load(k_ptrs, mask=(start_n + offs_n[:, None]) < seqlen, other=0.0)
        qk = tl.dot(q, tl.trans(k))
        qk = tl.where(offs_m[:, None] >= (start_n + offs_n[None, :]), qk, float("-inf"))
        m_ij = tl.max(qk, axis=1)
        m_new = tl.maximum(m_i, m_ij)
        alpha = tl.math.exp2(m_i - m_new)
        p = tl.math.exp2(qk - m_new[:, None])
        l_i = l_i * alpha + tl.sum(p, axis=1)
        acc = acc * alpha[:, None]
        v_ptrs = V + v_offset + (start_n + offs_n[:, None]) * stride_vn + offs_d[None, :] * stride_vk
 ... 8 lines elided ...


def custom_attention_forward(q, k, v, causal=True, sm_scale=None):
    """Optimized wrapper: fuse log2(e)*sm_scale into Q, autotuned kernel launch."""
    batch, nheads, seqlen, headdim = q.shape
    q, k, v = q.contiguous(), k.contiguous(), v.contiguous()
    if sm_scale is None:
        sm_scale = 1.0 / math.sqrt(headdim)
    # Fuse sm_scale * log2(e) into Q to avoid per-element multiply in inner loop
    q = (q * (sm_scale * 1.44269504)).contiguous()
    o = torch.empty_like(q)
    grid = lambda META: (triton.cdiv(seqlen, META['BLOCK_M']), batch * nheads)
    _custom_attn_fwd[grid](
        q, k, v, o,
        q.stride(1), q.stride(2), q.stride(3),
        k.stride(1), k.stride(2), k.stride(3),
        v.stride(1), v.stride(2), v.stride(3),
```

Expert Assessment. Reproduces the causal / non-causal split correctly and adds a fairly comprehensive set of autotune configurations targeted at H100. Recovers Flash-Attention v2's delayed block-wise division. Pre-scales Q outside the kernel, which is a wash in this regime. Did not adopt Hopper-specific primitives such as TMA. Overall an FA2-class implementation, well-tuned for H100 but not Hopper-native.

Agent: Gemini 3.1 Pro.
TMA-style attempt via `tl.make_block_ptr`.

```
27 # ==============================================================
28
 - editable region begins at line 29 -
29 @triton.autotune(
30     configs=[
31         triton.Config({'BLOCK_M': 256, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
32         triton.Config({'BLOCK_M': 256, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
33         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 256}, num_stages=3, num_warps=8),
34         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
35         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=4, num_warps=8),
36         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 128}, num_stages=5, num_warps=8),
37         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
38         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=4, num_warps=8),
39         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=5, num_warps=8),
40         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 256}, num_stages=3, num_warps=8),
41         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 256}, num_stages=4, num_warps=8),
42         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 128}, num_stages=3, num_warps=8),
43         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 128}, num_stages=4, num_warps=8),
44         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 128}, num_stages=5, num_warps=8),
45         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=3, num_warps=4),
46         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=4, num_warps=4),
47         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=5, num_warps=4),
48         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=4, num_warps=8),
49         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=5, num_warps=8),
50         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=8, num_warps=4),
51         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 32}, num_stages=4, num_warps=4),
52         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 32}, num_stages=5, num_warps=8),
53         # Extra configs specifically for D=256
54         triton.Config({'BLOCK_M': 128, 'BLOCK_N': 64}, num_stages=2, num_warps=8),
55         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=3, num_warps=8),
56         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=2, num_warps=4),
57         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 64}, num_stages=2, num_warps=8),
58         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 32}, num_stages=2, num_warps=4),
59         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 32}, num_stages=3, num_warps=4),
60         triton.Config({'BLOCK_M': 64, 'BLOCK_N': 32}, num_stages=4, num_warps=4),
 ... 88 lines elided ...
164         m_i = m_new
165
166         k_block_ptr = tl.advance(k_block_ptr, (0, BLOCK_N))
167         v_block_ptr = tl.advance(v_block_ptr, (BLOCK_N, 0))
168
169     acc = acc / l_i[:, None]
170
171     o_block_ptr = tl.make_block_ptr(
172         base=Out + o_offset,
173         shape=(seqlen, BLOCK_DMODEL),
174         strides=(stride_om, stride_ok),
175         offsets=(start_m * BLOCK_M, 0),
176         block_shape=(BLOCK_M, BLOCK_DMODEL),
177         order=(1, 0),
178     )
179     tl.store(o_block_ptr, acc.to(Out.dtype.element_ty))
180
181 def custom_attention_forward(q, k, v, causal=True, sm_scale=None):
182     batch, nheads, seqlen, headdim = q.shape
183     q, k, v = q.contiguous(), k.contiguous(), v.contiguous()
184     if sm_scale is None:
185         sm_scale = 1.0 / math.sqrt(headdim)
186
187     scale = sm_scale * 1.44269504
188     o = torch.empty_like(q)
189
190     grid = lambda META: (triton.cdiv(seqlen, META['BLOCK_M']), batch * nheads)
191
192     _custom_attn_fwd[grid](
193         q, k, v, o,
194         scale,
195         q.stride(1), q.stride(2), q.stride(3),
196         k.stride(1), k.stride(2), k.stride(3),
197         v.stride(1), v.stride(2), v.stride(3),
```

Expert Assessment. The most interesting attempt at Hopper-specific optimisation. Uses `tl.make_block_ptr` *to express the K/V/O loads, which is the Triton-level idiom that maps onto the H100's tensor-memory accelerator. Otherwise an FA2 variant: causal split, autotune configs, delayed denominator division, in-kernel scaling of Q. A reasonable preliminary step toward Hopper-native attention.*

Agent: Qwen 3.6 Plus.

A failure case (kept for contrast).

```
    BLOCK_DMODEL: tl.constexpr,
    IS_CAUSAL: tl.constexpr,
):
    """Optimized fused self-attention: two-pass causal + fused scale."""
    start_m = tl.program_id(0)
    off_hz = tl.program_id(1)

... 6 lines elided ...
    offs_n = tl.arange(0, BLOCK_N)
    offs_d = tl.arange(0, BLOCK_DMODEL)

    # Load Q tile (scale pre-fused in wrapper with log2(e) factor)
    q_ptrs = Q + q_offset + offs_m[:, None] * stride_qm + offs_d[None, :] * stride_qk
    q = tl.load(q_ptrs, mask=offs_m[:, None] < seqlen, other=0.0)

... 2 lines elided ...
    l_i = tl.zeros([BLOCK_M], dtype=tl.float32)
    acc = tl.zeros([BLOCK_M, BLOCK_DMODEL], dtype=tl.float32)

    # --- Pass 1: Non-causal blocks (no masking needed, full pipelining) ---
    if IS_CAUSAL:
        non_causal_end = (start_m * BLOCK_M // BLOCK_N) * BLOCK_N
    else:
        non_causal_end = seqlen

    for start_n in range(0, non_causal_end, BLOCK_N):
        start_n = tl.multiple_of(start_n, BLOCK_N)
        k_ptrs = K + k_offset + (start_n + offs_n[:, None]) * stride_kn + offs_d[None, :] * stride_kk
        k = tl.load(k_ptrs)
        qk = tl.dot(q, tl.trans(k))
        m_ij = tl.max(qk, axis=1)
        m_new = tl.maximum(m_i, m_ij)
        alpha = tl.math.exp2(m_i - m_new)
        p = tl.math.exp2(qk - m_new[:, None])
        l_i = l_i * alpha + tl.sum(p, axis=1)
... 38 lines elided ...
        sm_scale = 1.0 / math.sqrt(headdim)
    # Fuse softmax scale and log2(e) into Q
    q = (q * (sm_scale * 1.4426950408889634)).contiguous()
    o = torch.empty_like(q)

    # Per-headdim tuned configurations for H100
    if headdim <= 64:
        # Small head: memory-bound, use large blocks to amortize overhead
        BLOCK_M, BLOCK_N = 128, 64
        num_stages, num_warps = 4, 8
    elif headdim <= 128:
        # Medium head: balanced
        BLOCK_M, BLOCK_N = 128, 64
        num_stages, num_warps = 4, 8
    else:
        # Large head (256): compute-bound, use larger blocks for tensor core efficiency
        BLOCK_M, BLOCK_N = 64, 64
        num_stages, num_warps = 4, 8

    grid = (triton.cdiv(seqlen, BLOCK_M), batch * nheads)
    _custom_attn_fwd[grid](
        q, k, v, o,
        q.stride(1), q.stride(2), q.stride(3),
        k.stride(1), k.stride(2), k.stride(3),
        v.stride(1), v.stride(2), v.stride(3),
... 1 lines elided ...
        seqlen,
```

```
148         BLOCK_M=BLOCK_M, BLOCK_N=BLOCK_N,
149         BLOCK_DMODEL=headdim, IS_CAUSAL=causal,
150         num_stages=num_stages, num_warps=num_warps,
151     )
152     return o
153
 - editable region ends at line 153 -
```

Expert Assessment. Effectively a failed attempt: passes the correctness check but does not realise meaningful speedup over the SDPA baseline. Useful as a contrast: even when the editable region is small and the surrounding harness fixes the evaluation protocol, a weak kernel cannot recover throughput.

F.2. L_∞ Adversarial Training for Robust Accuracy

Task. A custom adversarial training procedure is evaluated on robust accuracy under white-box ℓ_∞ attacks while preserving clean accuracy across MNIST, CIFAR-10, and CIFAR-100 settings. Editable: the AdversarialTrainer inner attack and outer training loss in `custom_adv_train.py`. Read-only: the data loaders, model architectures, optimizer, learning-rate schedule, and evaluation attacks. Provided baselines are standard training, PGD-AT, TRADES, MART, and AWP.

```
10 class AdversarialTrainer:
11     """
12     Adversarial training method.
13
14     The agent should modify this class to implement a better adversarial
15     training procedure that improves model robustness against L_inf attacks.
16
17     Args:
18         model (nn.Module): The model to train.
19         eps (float): L_inf perturbation budget.
20         alpha (float): Step size for adversarial perturbation generation.
21         attack_steps (int): Number of PGD steps for adversarial example generation.
 ... 20 lines elided ...
42             dict: Must contain 'loss' key (float).
43         """
44         # Default: standard (non-adversarial) training
45         self.model.train()
46         outputs = self.model(images)
47         loss = F.cross_entropy(outputs, labels)
48
49         optimizer.zero_grad()
50         loss.backward()
51         optimizer.step()
52
53         return {'loss': loss.item()}
54
```

```
 8 # EDITABLE -- implement AdversarialTrainer below
 9 # ================================================================
  - editable region begins at line 10 -
10 class AdversarialTrainer:
11     """PGD Adversarial Training (Madry et al., 2018)."""
12
13     def __init__(self, model, eps, alpha, attack_steps, num_classes, **kwargs):
14         self.model = model
 ... 3 lines elided ...
18         self.num_classes = num_classes
19
20     def train_step(self, images, labels, optimizer):
21         # Generate adversarial examples using PGD
22         self.model.eval()
23         adv_images = images.clone().detach()
24         adv_images = adv_images + torch.empty_like(adv_images).uniform_(-self.eps, self.eps)
25         adv_images = torch.clamp(adv_images, 0.0, 1.0)
26
```

```
        for _ in range(self.attack_steps):
            adv_images.requires_grad_(True)
            outputs = self.model(adv_images)
            loss = F.cross_entropy(outputs, labels)
            grad = torch.autograd.grad(loss, adv_images)[0]
            adv_images = adv_images.detach() + self.alpha * grad.sign()
            delta = torch.clamp(adv_images - images, min=-self.eps, max=self.eps)
            adv_images = torch.clamp(images + delta, 0.0, 1.0).detach()

        # Train on adversarial examples
        self.model.train()
        outputs = self.model(adv_images)
        loss = F.cross_entropy(outputs, labels)

        optimizer.zero_grad()
```

Agent: GPT-5.4.

PGD with margin-gap inner loss; logit-margin-aware outer weights.

```
# EDITABLE -- implement AdversarialTrainer below
# ================================================================
 - editable region begins at line 10 -
class AdversarialTrainer:
    """Adaptive Anchor-Margin Adversarial Training (AAMAT).

    Novelty:
    1) adversarial examples maximize CE + margin-collapse loss;
    2) adversarial supervision uses a confidence-adaptive anchor target that
       interpolates between one-hot labels and the clean predictive distribution;
    3) robust regularization is upweighted for low-margin / low-confidence samples.
    """

    def __init__(self, model, eps, alpha, attack_steps, num_classes, **kwargs):
 ... 2 lines elided ...
        self.alpha = alpha
        self.attack_steps = attack_steps
        self.num_classes = num_classes
        self.beta = 1.2
        self.gamma = 0.6
        self.lam_attack = 1.0

    def train_step(self, images, labels, optimizer):
        self.model.train()
        logits_clean = self.model(images)
        loss_clean = F.cross_entropy(logits_clean, labels)

        with torch.no_grad():
            probs = F.softmax(logits_clean, dim=1)
            p_true = probs.gather(1, labels.unsqueeze(1)).squeeze(1)
            masked = logits_clean.detach().clone()
            masked.scatter_(1, labels.unsqueeze(1), -1e9)
            top_other = masked.max(dim=1).values
            true_logit = logits_clean.detach().gather(1, labels.unsqueeze(1)).squeeze(1)
            margin = true_logit - top_other
 ... 6 lines elided ...

        self.model.eval()
        adv_images = torch.clamp(images.detach() + 0.001 * torch.randn_like(images), 0.0, 1.0)
        for _ in range(self.attack_steps):
            adv_images.requires_grad_(True)
            logits_adv = self.model(adv_images)
            true_adv = logits_adv.gather(1, labels.unsqueeze(1)).squeeze(1)
            masked_adv = logits_adv.clone()
            masked_adv.scatter_(1, labels.unsqueeze(1), -1e9)
            max_other_adv = masked_adv.max(dim=1).values
            loss_attack = F.cross_entropy(logits_adv, labels) + self.lam_attack * F.softplus(max_other_adv - true_adv).mean()
            grad = torch.autograd.grad(loss_attack, adv_images)[0]
            adv_images = adv_images.detach() + self.alpha * grad.sign()
```

```
62            delta = torch.clamp(adv_images - images, min=-self.eps, max=self.eps)
63            adv_images = torch.clamp(images + delta, 0.0, 1.0).detach()
64
65        self.model.train()
66        logits_adv = self.model(adv_images)
67        adv_logp = F.log_softmax(logits_adv, dim=1)
68        loss_anchor = (weight * (-(anchor * adv_logp).sum(dim=1))).mean()
69        true_adv = logits_adv.gather(1, labels.unsqueeze(1)).squeeze(1)
70        masked_adv = logits_adv.clone()
71        masked_adv.scatter_(1, labels.unsqueeze(1), -1e9)
72        max_other_adv = masked_adv.max(dim=1).values
73        loss_margin = (weight * F.softplus(max_other_adv - true_adv)).mean()
74        loss = loss_clean + self.beta * loss_anchor + self.gamma * loss_margin
75
76        optimizer.zero_grad()
77        loss.backward()
78        optimizer.step()
79        return {'loss': loss.item(), 'loss_clean': loss_clean.item(), 'loss_anchor': loss_anchor.item()}
 - editable region ends at line 79 -
80 # ==================================================================
81 # END EDITABLE
```

Expert Assessment. The most distinctive method among the five. Inner attack maximises $\mathrm{CE} + \lambda \cdot (\textit{margin-gap loss})$*, and the outer loss reweights examples as* $0.35 + 0.65 \cdot \max\big(1 - p_y,\ \sigma(-\textit{margin})\big)$*. It is also the model with the shortest reasoning trace by far (only two iterations with thinking, around 5 000 characters total). Only GPT considered the logit margin explicitly.*

Agent: Qwen 3.6 Plus.
TRADES-style attack with focal-weighted outer loss.

```
 9 # ==================================================================
  - editable region begins at line 10 -
10 class AdversarialTrainer:
11     """
12     Robust Focal TRADES (RFT).
13
14     Novel contributions:
15     1. Dual-objective: Clean CE preserves natural accuracy while
16        adversarial CE directly trains robust classification on perturbed inputs.
17     2. Confidence-adaptive focal weighting: both adversarial CE and KL
18        divergence are weighted by (1 - p_adv)^gamma, dynamically prioritizing
19        samples where the model is currently vulnerable on adversarial inputs.
20        This generalizes MART (clean-confidence-weighted) to use the adversarial
21        prediction confidence as the weighting signal.
22     3. KL divergence regularizer preserves decision boundary smoothness between
23        clean and adversarial predictions.
24
25     Key design choices:
26     - gamma=1.0: balanced emphasis. gamma=0 recovers standard TRADES + PGDAT.
27     - Adversarial CE uses focal weighting to target the hardest adversarial
28       examples directly.
29     - KL regularization prevents overconfident predictions and maintains
30       decision boundary smoothness.
31     """
32
33     def __init__(self, model, eps, alpha, attack_steps, num_classes, **kwargs):
 ... 2 lines elided ...
36         self.alpha = alpha
37         self.attack_steps = attack_steps
38         self.num_classes = num_classes
39         self.beta = 6.0     # KL regularization weight (TRADES-style)
40         self.gamma = 1.0    # Focal exponent for confidence-adaptive weighting
41
42     def train_step(self, images, labels, optimizer):
43         # == Step 1: Generate adversarial examples via KL maximization ==
 ... 35 lines elided ...
79         loss_clean = F.cross_entropy(logits_clean, labels)
80
81         # Component B: Focal-weighted adversarial CE
82         # Directly improves robust classification on perturbed inputs
83         ce_adv = F.cross_entropy(logits_adv, labels, reduction='none')
```

```
84          loss_adv = (focal_weight * ce_adv).mean()
85
86          # Component C: Focal-weighted KL divergence (smoothness regularization)
87          # KL(adv || clean) weighted by adversarial focal weight
88          nat_probs = F.softmax(logits_clean, dim=1)
89          kl_per_sample = F.kl_div(
90              torch.log(adv_probs + 1e-12),
91              nat_probs,
92              reduction='none',
93          ).sum(dim=1)
94          loss_kl = (focal_weight * kl_per_sample).mean()
95
96          # Combined: clean + adv focal + KL regularizer
97          loss = loss_clean + loss_adv + self.beta * loss_kl
98
99          optimizer.zero_grad()
100          loss.backward()
101          optimizer.step()
102
103          return {
104              'loss': loss.item(),
105              'loss_clean': loss_clean.item(),
106              'loss_adv': loss_adv.item(),
107              'loss_kl': loss_kl.item(),
108          }
109
 - editable region ends at line 109 -
110 # ==================================================================
111 # END EDITABLE
```

Expert Assessment. TRADES-style PGD inner attack combined with a focal-weighted outer loss $L = \mathrm{CE}_{clean} + w \cdot \mathrm{CE}_{adv} + 6w \cdot \mathrm{KL}$*, where* $w = (1 - p_{adv})$*. Performs worst on MNIST among all five models, but best on both CIFAR-10 and CIFAR-100. The split is informative: the focal weighting helps on harder, multi-class problems but destabilises the easy MNIST regime.*

F.3. Quantization-Aware Language-Model Training

Task. A training-side quantization-aware training algorithm is evaluated by the WikiText-2 perplexity gap between full-precision Pythia-1.4B and INT4, INT3, and INT2 group-quantized variants. Editable: the fake-quant forward, gradient surrogate, real quantize-dequantize path, QAT wrapper, learnable parameters, and `CONFIG_OVERRIDES` in `custom_qat.py`. Read-only: model loading, WikiText-2 sampling, the training loop, final real-QDQ roundtrip, and perplexity evaluation. Provided baselines are `no_qat`, `ste`, `lsq`, and `finetune_then_ptq`.

```
33 # Per-method training hyperparameters.  The training loop reads this dict.
34 # Override any of these in your method to retune.
35 CONFIG_OVERRIDES = {
36     "learning_rate": 2e-5,
37     "num_steps": 500,
38     "batch_size": 2,
39     "gradient_accumulation_steps": 4,
40     "max_grad_norm": 1.0,
41     "warmup_steps": 50,
42     "weight_decay": 0.0,
43 }
44
 ... 119 lines elided ...
164             else:
165                 _replace(child)
166
167     _replace(model)
168     # Restore the LM head to full precision (covers GPT-2 `lm_head` and
169     # Pythia / GPTNeoX `embed_out`).
170     for head_attr in ("lm_head", "embed_out"):
171         head = getattr(model, head_attr, None)
172         if isinstance(head, QATWrapper):
173             setattr(model, head_attr, head.linear)
174
175     return model
```

```
# ==============================================================================

 - editable region begins at line 33 -

# == Straight-Through Estimator (STE) QAT baseline ============================

CONFIG_OVERRIDES = {
    "learning_rate": 2e-5,
    "num_steps": 500,
 ... 14 lines elided ...
def fake_quantize_weight(weight, num_bits, group_size):
    qmin, qmax = _qrange(num_bits)
    out_features, in_features = weight.shape
    assert in_features % group_size == 0
    w = weight.float().reshape(out_features, -1, group_size)
    # Recompute scale on-the-fly each forward (max-abs / qmax).
    w_max = w.abs().amax(dim=-1, keepdim=True).clamp(min=1e-12)
    scale = w_max / qmax
    w_q = torch.clamp(torch.round(w / scale), qmin, qmax) * scale
    # Straight-through: forward = quantized, backward = identity.
    w_dq = w + (w_q - w).detach()
    return w_dq.reshape(out_features, in_features).to(weight.dtype)

 ... 36 lines elided ...


def prepare_qat_model(model, num_bits, group_size):
    from transformers.pytorch_utils import Conv1D

    def _replace(parent):
        for name, child in list(parent.named_children()):
            if isinstance(child, nn.Linear):
                setattr(parent, name, QATWrapper(child, num_bits=num_bits, group_size=group_size))
            elif isinstance(child, Conv1D):
                in_f, out_f = child.weight.shape
                lin = nn.Linear(in_f, out_f, bias=child.bias is not None,
 ... 2 lines elided ...
                    lin.weight.copy_(child.weight.t().contiguous())
                    if child.bias is not None:
                        lin.bias.copy_(child.bias)
                setattr(parent, name, QATWrapper(lin, num_bits=num_bits, group_size=group_size))
            else:
                _replace(child)

```

Agent: GPT-5.4.

RTN-anchored LSQ with cosine schedule between learned grid and RTN.

```
    return qmin, qmax


class _LSQQuant(torch.autograd.Function):
    @staticmethod
    def forward(ctx, w, scale, qmin, qmax, g_scale):
        w_div = w / scale
        w_clip = torch.clamp(w_div, qmin, qmax)
        w_round = torch.round(w_clip)
        ctx.save_for_backward(w_div, scale)
        ctx.qmin = qmin
        ctx.qmax = qmax
        ctx.g_scale = g_scale
        return w_round * scale

    @staticmethod
```

```
    def backward(ctx, grad_out):
        w_div, scale = ctx.saved_tensors
        qmin, qmax, g = ctx.qmin, ctx.qmax, ctx.g_scale
        in_range = (w_div > qmin) & (w_div < qmax)
        grad_w = torch.where(in_range, grad_out, torch.zeros_like(grad_out))
        below = (w_div <= qmin).float() * float(qmin)
        above = (w_div >= qmax).float() * float(qmax)
        inside = in_range.float() * (torch.round(w_div) - w_div)
        grad_s = ((below + above + inside) * grad_out).sum(dim=-1, keepdim=True) * g
        return grad_w, grad_s, None, None, None


def _reshape_groups(weight, group_size):
    out_features, in_features = weight.shape
    assert in_features % group_size == 0, (
        f"in_features {in_features} not divisible by group_size {group_size}"
    )
    return weight.float().reshape(out_features, -1, group_size)

 ... 81 lines elided ...

    def forward(self, x):
        x = fake_quantize_activation(x, self.num_bits)
        w_q = fake_quantize_weight(
            self.linear.weight,
            self.num_bits,
            self.group_size,
            scale=self._lsq_scale(),
            blend=self._blend(),
        )
        return F.linear(x, w_q, self.linear.bias)

    def aux_loss(self, step, total_steps):
        self._step.fill_(int(step))
        self._total_steps.fill_(int(total_steps))
        p = self._progress()
        qmin, qmax = _qrange(self.num_bits)
        w = _reshape_groups(self.linear.weight, self.group_size)
        w_max = w.abs().amax(dim=-1, keepdim=True).clamp(min=1e-12)
        rtn_scale = w_max / max(1, qmax)
        scaled = torch.clamp(w / rtn_scale, qmin, qmax)
        frac = scaled - torch.round(scaled)
        # Focus on values that are not already near a clipping boundary.
        center_mask = (scaled.abs() < (qmax - 0.25)).float()
        align = (frac.pow(2) * center_mask).sum() / center_mask.sum().clamp(min=1.0)
        # Keep learned LSQ scale near RTN scale so the fake-quant grid does not drift.
        scale_reg = ((self._lsq_scale() - rtn_scale) / rtn_scale.clamp(min=1e-8)).pow(2).mean()
        late = p * p
        align_w = (0.0015 if self.num_bits >= 4 else 0.003 if self.num_bits == 3 else 0.006) * late
        scale_w = 0.0005
        return align_w * align + scale_w * scale_reg


def prepare_qat_model(model, num_bits, group_size):
```

Expert Assessment. During QAT the method interpolates between a deployment RTN grid and a learned LSQ grid using a cosine schedule that stays closer to RTN. Conceptually a clean attempt: a hybrid that keeps PTQ-deployable scales while letting the network adapt. However, the final evaluation ignores the learned scale and recomputes pure RTN max-abs from the trained weights, so most of the QAT work is discarded at deployment time. The INT2/3/4 results are all weak; without a controlled ablation it is unclear whether the underlying method is at fault or the eval-time RTN re-scaling.

Agent: Claude Opus 4.6.

Adaptive clipping + noise injection + Stable-QAT EMA + AdaRound aux.

```
# ==============================================================================

 - editable region begins at line 33 -

# == Adaptive Clipping QAT with Gradual Noise Injection (AC-GNI) v2 ===========
```

```
# Key improvements over v1:
# 1) Bit-adaptive clipping init: more aggressive clipping at lower bits
#    (clip_ratio = 0.80 at INT2 vs 0.95 at INT4)
# 2) Extended noise warmup: 50% of training at INT2, 30% at INT3/4
# 3) EMA scale smoothing: exponential moving average on learned scales
#    prevents scale oscillation that causes INT2 divergence
# 4) Stronger outlier regularization with annealing: start strong, decay
# 5) Quantization-error-aware loss: small MSE term between FP and quantized
#    weights during training to keep weights near grid points

CONFIG_OVERRIDES = {
    "learning_rate": 2e-5,
    "num_steps": 500,
... 12 lines elided ...
    return qmin, qmax


class _ACGNIQuantFn(torch.autograd.Function):
    """Custom autograd: LSQ-style quant with learned clipping bounds."""

    @staticmethod
    def forward(ctx, w, scale, qmin, qmax, grad_scale_factor):
        w_div = w / scale
        w_clip = torch.clamp(w_div, qmin, qmax)
        w_round = torch.round(w_clip)
        ctx.save_for_backward(w_div, scale)
        ctx.qmin = qmin
        ctx.qmax = qmax
        ctx.grad_scale_factor = grad_scale_factor
        return w_round * scale
... 171 lines elided ...
            qe_strength = 0.001
            if self.num_bits <= 2:
                qe_strength = 0.005
            outlier_reg = outlier_reg + qe * qe_strength

        return outlier_reg


def prepare_qat_model(model, num_bits, group_size):
    """Replace nn.Linear layers with QATWrapper (LM head stays FP)."""
    from transformers.pytorch_utils import Conv1D

    QATWrapper._num_bits_global = num_bits

    def _replace(parent):
        for name, child in list(parent.named_children()):
            if isinstance(child, nn.Linear):
                wrapper = QATWrapper(child, num_bits=num_bits,
                                     group_size=group_size)
                setattr(parent, name, wrapper)
            elif isinstance(child, Conv1D):
                in_f, out_f = child.weight.shape
                lin = nn.Linear(in_f, out_f, bias=child.bias is not None,
                                device=child.weight.device,
                                dtype=child.weight.dtype)
                with torch.no_grad():
                    lin.weight.copy_(child.weight.t().contiguous())
                    if child.bias is not None:
                        lin.bias.copy_(child.bias)
                wrapper = QATWrapper(lin, num_bits=num_bits,
                                     group_size=group_size)
                setattr(parent, name, wrapper)
            else:
                _replace(child)
```

Expert Assessment. A combinatorial attempt: adaptive clipping range, gradual noise injection, Stable-QAT-style EMA, plus an AdaRound-style auxiliary loss. Considerable hyperparameter tuning around the noise schedule and clip ranges, but each component is taken off-the-shelf. We do not see strong methodological novelty beyond the combination itself.

F.4. Latent Normalization for World Models

Task. A custom latent normalization layer is evaluated inside the TD-MPC2 world model encoder and dynamics network, with episode reward measured on DMControl walker-walk, cheetah-run, and a hidden cartpole-swingup split. Editable: the CustomSimNorm class in `custom_simnorm.py`. Read-only: the surrounding encoder, dynamics model, training procedure, and evaluation harness. Provided baselines are SimNorm, L2Norm, RMSNorm, and identity normalization.

```
16 class CustomSimNorm(nn.Module):
17     """Custom normalization for latent state representations in world models.
18
19     Interface contract (same as SimNorm):
20         __init__(cfg)  -- cfg.simnorm_dim is the group size (default: 8)
21         forward(x: Tensor) -> Tensor  (same shape as input)
22
23     The input tensor has shape (*batch_dims, latent_dim) where latent_dim
24     is divisible by simnorm_dim. Your normalization should constrain the
25     geometry of the latent space to improve world model learning.
26
27     Evaluated on DMControl walker-walk and cheetah-run tasks.
... 3 lines elided ...
31         super().__init__()
32         self.dim = cfg.simnorm_dim
33
34     def forward(self, x):
35         # Default: SimNorm (simplicial normalization)
36         # Reshape into groups of size self.dim and apply softmax
37         shp = x.shape
38         x = x.view(*shp[:-1], -1, self.dim)
39         x = F.softmax(x, dim=-1)
40         return x.view(*shp)
41
42     def __repr__(self):
43         return f"CustomSimNorm(dim={self.dim})"
```

```
14 # EDITABLE: Custom latent normalization
15 # ======================================================================
- editable region begins at line 16 -
16 class CustomSimNorm(nn.Module):
17     """L2 normalization baseline -- projects latent vectors onto a hypersphere."""
18
19     def __init__(self, cfg):
20         super().__init__()
21         self.dim = cfg.simnorm_dim
22         self.eps = 1e-8
23         # Learnable scale parameter
24         self.scale = nn.Parameter(torch.ones(1))
25
26     def forward(self, x):
27         # L2 normalize across the last dimension and apply learnable scale
28         norm = torch.norm(x, p=2, dim=-1, keepdim=True).clamp(min=self.eps)
29         return self.scale * x / norm
30
31     def __repr__(self):
32         return f"CustomSimNorm(dim={self.dim}, type=L2Norm)"
- editable region ends at line 32 -
```

Agent: GPT-5.4.

Entmax-1.5-gated spherical simplex normalization.

```
14 # EDITABLE: Custom latent normalization
15 # ======================================================================
- editable region begins at line 16 -
16 class CustomSimNorm(nn.Module):
17     """Entmax-gated spherical simplex normalization.
```

```
18
19      Novel contribution:
20      - Group-wise temperature-free sparse competition via entmax15 instead of
21        softmax, yielding adaptive sparsity within each latent simplex group.
22      - Residual signed direction path preserving magnitude-free angular
23        information via per-group L2 normalization.
24      - Data-dependent interpolation between simplex and spherical views using
25        an intrinsic confidence score from the group logits.
26
27      This aims to combine SimNorm's stable bounded geometry with richer signed
28      relational information useful for latent dynamics prediction.
29      """
30
31      def __init__(self, cfg):
32          super().__init__()
33          self.dim = cfg.simnorm_dim
34          self.eps = 1e-8
35          # Hardcoded coefficients per instructions.
36          self.mix = 0.35
37          self.conf_scale = 1.25
38
39      def _entmax15(self, x):
40          # Exact entmax with alpha=1.5 via bisection; group size is tiny (8), so
41          # this is computationally cheap and numerically stable.
42          max_val = x.max(dim=-1, keepdim=True).values
43          x = x - max_val
44          lo = x.min(dim=-1, keepdim=True).values - 1.0
45          hi = x.max(dim=-1, keepdim=True).values
46          for _ in range(12):
47              tau = (lo + hi) / 2.0
48              p = torch.clamp((x - tau) / 2.0, min=0.0).pow(2)
49              s = p.sum(dim=-1, keepdim=True)
50              lo = torch.where(s > 1.0, tau, lo)
51              hi = torch.where(s <= 1.0, tau, hi)
52          tau = hi
53          p = torch.clamp((x - tau) / 2.0, min=0.0).pow(2)
54          p = p / p.sum(dim=-1, keepdim=True).clamp_min(self.eps)
55          return p
56
57      def forward(self, x):
58          shp = x.shape
59          xg = x.view(*shp[:-1], -1, self.dim)
60
61          simplex = self._entmax15(xg)
62          centered = xg - xg.mean(dim=-1, keepdim=True)
63          sphere = centered / torch.norm(centered, p=2, dim=-1, keepdim=True).clamp_min(self.eps)
64
65          confidence = (xg.max(dim=-1, keepdim=True).values - xg.mean(dim=-1, keepdim=True)).sigmoid()
66          alpha = self.mix * (confidence * self.conf_scale).clamp(max=1.0)
67
68          y = (1.0 - alpha) * simplex + alpha * sphere
69          return y.view(*shp)
70
71      def __repr__(self):
72          return f"CustomSimNorm(dim={self.dim}, type=EntmaxSphericalMix)"
 - editable region ends at line 72 -
```

Expert Assessment. The most structurally different attempt. Replaces the per-group softmax with sparse entmax-1.5, then mixes in a signed per-group ℓ_2-normalized spherical path with a confidence-dependent coefficient. Reads more as a recombination of pre-existing techniques (entmax from attention literature, spherical normalization from contrastive learning) than a method with a clear hypothesis about world models. Does not lead to consistent gains.

Agent: Gemini 3.1 Pro.

Hellinger / Gated SimNorm.

```
14 # EDITABLE: Custom latent normalization
15 # ==================================================================
 - editable region begins at line 16 -
16 class CustomSimNorm(nn.Module):
```

```
    """Gated Simplicial Normalization (GatedSimNorm).
    Reclaims the shift-invariance degree of freedom in standard SimNorm
    by using the mean logit of each group to compute a Sigmoid gate.
    This allows the world model to dynamically turn off inactive latent
    groups, increasing representation capacity and sparsity, while strictly
    bounding the latent space to prevent Q-value explosion.
    """
    def __init__(self, cfg):
        super().__init__()
        self.dim = cfg.simnorm_dim
        # Global bias to ensure groups are active by default at initialization
        self.gate_bias = nn.Parameter(torch.tensor(2.0))

    def forward(self, x):
        shp = x.shape
        x = x.view(*shp[:-1], -1, self.dim)

        # Categorical distribution (depends only on x - mean(x))
        sim = F.softmax(x, dim=-1)

        # Group activation gate (depends only on mean(x))
        gate = torch.sigmoid(x.mean(dim=-1, keepdim=True) + self.gate_bias)

        x = sim * gate
        return x.view(*shp)

    def __repr__(self):
        return f"CustomSimNorm(dim={self.dim}, type=GatedSimNorm)"
 - editable region ends at line 44 -
```

Expert Assessment. First tries to enforce a spherical latent distribution via a Hellinger-distance term — a non-trivial geometric idea mapping softmax probabilities to a sphere-like manifold — and then switches to a Gated SimNorm that lets some latent groups stay inactive in order to avoid Q-value blow-up downstream. Both ideas are reasonably novel for this setting, though the motivation linking the geometry change to world model performance is not made explicit, and neither variant consistently outperforms the baselines.

Agent: Qwen 3.6 Plus.

Residual L2 / SimNorm blend with learned per-group weights.

```
# EDITABLE: Custom latent normalization
# ======================================================================
 - editable region begins at line 16 -
class CustomSimNorm(nn.Module):
    """Centered Adaptive Hybrid Normalization with Group-wise Learned Weights.

    Key ideas:
    1. Mean-centering within groups to remove bias before normalization
    2. Two parallel paths: group-wise L2 and temperature-controlled SimNorm
    3. Learnable blending between simplex and hypersphere geometry
    4. Per-element learned weights for fine-grained control
    5. Global L2 normalization for final stability

    Mean-centering helps by ensuring the normalization operates on meaningful
    signal rather than being dominated by group means.
    """

    def __init__(self, cfg):
        super().__init__()
        self.dim = cfg.simnorm_dim
        self.eps = 1e-8

        # Learnable mixing coefficient
        self.alpha = nn.Parameter(torch.ones(1) * 0.6)

        # Temperature for softmax
        self.temperature = nn.Parameter(torch.ones(1) * 1.0)

        # Per-element learned weight
        self.weight = nn.Parameter(torch.ones(self.dim))

```

```
44         # Learnable scale factor
45         self.scale = nn.Parameter(torch.ones(1) * 1.0)
46
47         # Weight for centering: how much mean to subtract
 ... 4 lines elided ...
52         shp = x.shape
53         x = x.view(*shp[:-1], -1, self.dim)
54
55         # Mean-centering within each group
56         center_w = torch.sigmoid(self.center_weight)
57         x = x - center_w * x.mean(dim=-1, keepdim=True)
58
59         # Path 1: Group-wise L2 normalization
60         norm_l2 = torch.norm(x, p=2, dim=-1, keepdim=True).clamp(min=self.eps)
61         x_l2 = x / norm_l2
62
63         # Path 2: Temperature-controlled SimNorm
64         temp = self.temperature.clamp(min=0.1, max=10.0)
65         x_sim = F.softmax(x / temp, dim=-1)
66
67         # Blend between paths
68         alpha = torch.sigmoid(self.alpha)
69         out = alpha * x_sim + (1.0 - alpha) * x_l2
70
71         # Apply per-element learned weights
72         out = out * self.weight
73
74         # Reshape back
75         out = out.view(*shp)
76
77         # Global L2 normalization for stability
78         norm_global = torch.norm(out, p=2, dim=-1, keepdim=True).clamp(min=self.eps)
79         out = out / norm_global
80
81         return out * self.scale
82
83     def __repr__(self):
84         return f"CustomSimNorm(dim={self.dim}, type=CenteredHybrid, alpha={torch.sigmoid(self.alpha).item():.3f})"
 - editable region ends at line 84 -
```

Expert Assessment. Adaptive hybrid normalization: residual blending of ℓ_2 normalisation with temperature-controlled SimNorm, plus learned per-group weights. The novelty is limited (each ingredient is pre-existing), but the empirical result is the cleanest of the five — visible gains on cheetah-run — so it is included as a contrast: a plain combination that nonetheless works.

F.5. Efficient Diffusion Sampling for Robot Actions

Task. A single inference-time sampler for a fixed DQL-style diffusion policy is evaluated on D4RL MuJoCo return at low numbers of function evaluations. Editable: the solver and `sampling_steps` fields in the custom MuJoCo YAML config. Read-only: the trained actor and critic, pipeline code, model architecture, training objective, datasets, environment list, seeds, evaluation loop, and NFE accounting. Provided baselines are DDPM, DDIM, and DPM-Solver++.

```
15 solver: ddpm
17 sampling_steps: 100
```

```
13
14 # Actor
 - editable region begins at line 15 -
15 solver: ode_dpmsolver++_2M
16 diffusion_steps: 100
17 sampling_steps: 10
 - editable region ends at line 17 -
18 predict_noise: True
19 ema_rate: 0.995
```

Agent: GPT-5.4.

Steps out of the inference-only box: critic-consistency action refinement.

```
    # Defines the actor (diffusion policy), optional critic(s), training loop,
    # and inference action-selection. The template defaults to Diffusion
    # Q-Learning (DQL): diffusion actor + twin Q critic with BC + Q loss.
    #
    # Novel contribution implemented here:
    #   1) Clipped advantage-matching actor regularization during training.
    #   2) Uncertainty-aware critic-guided elite refinement at inference.
    #   3) Barycentric fusion of refined elite actions, giving better actions at
    #      the same diffusion NFE budget.

    # --------------- Network Architecture -----------------
    nn_diffusion = DQLMlp(obs_dim, act_dim, emb_dim=64, timestep_emb_type="positional").to(args.device)
... 15 lines elided ...
    critic_target = deepcopy(critic).requires_grad_(False).eval()
    critic_optim = torch.optim.Adam(critic.parameters(), lr=args.critic_learning_rate)

    # ------------------ Algorithm knobs ---------------------
    q_margin_alpha = 0.20
    q_margin_clip = 5.0
    policy_value_alpha = 0.10

    # Inference-time refinement: no additional diffusion NFEs.
    refine_steps = 2
    refine_step_size = 0.05
    refine_topk = max(4, args.num_candidates // 5)
    elite_mix = 0.35
    uncertainty_weight = 0.50
    fusion_temperature = 6.0

    def conservative_q(q1, q2):
        q_mean = 0.5 * (q1 + q2)
        q_gap = torch.abs(q1 - q2)
        return q_mean - uncertainty_weight * q_gap

    # ---------------------- Training ----------------------
... 117 lines elided ...
                    refined_act = elite_act.view(args.num_envs, adaptive_topk, act_dim)

                    orig_best_idx = torch.argmax(base_util, dim=1)
                    orig_best_act = act_view[torch.arange(args.num_envs, device=args.device), orig_best_idx]
                    orig_best_util = base_util[torch.arange(args.num_envs, device=args.device), orig_best_idx].unsqueeze(-1)

                    blended_elite = ((1.0 - elite_mix) * refined_act + elite_mix * orig_best_act.unsqueeze(1)).clamp(-1.0, 1.0)
                    blended_q1, blended_q2 = critic_target(elite_obs, blended_elite.reshape(-1, act_dim))
                    blended_util = conservative_q(blended_q1, blended_q2).view(args.num_envs, adaptive_topk)

                    fusion_w = torch.softmax(refined_util * fusion_temperature, dim=1)
                    fused_act = (fusion_w.unsqueeze(-1) * refined_act).sum(dim=1).clamp(-1.0, 1.0)
                    fused_obs = obs_view[:, 0, :]
                    fused_q1, fused_q2 = critic_target(fused_obs, fused_act)
                    fused_util = conservative_q(fused_q1, fused_q2)

                    combined_util = torch.cat([refined_util, blended_util, orig_best_util, fused_util.unsqueeze(-1)], dim=1)
                    combined_act = torch.cat([
                        refined_act,
                        blended_elite,
                        orig_best_act.unsqueeze(1),
                        fused_act.unsqueeze(1)
                    ], dim=1)

                    final_idx = torch.argmax(combined_util, dim=1)
                    sampled_act = combined_act[torch.arange(args.num_envs, device=args.device), final_idx].cpu().numpy()

                prev_done = cum_done.copy()
                obs, rew, done, info = env_eval.step(sampled_act)
                ep_reward += rew * (1 - prev_done)
                cum_done = np.logical_or(cum_done, done)
                t += 1
```

```

            episode_rewards.append(ep_reward)
```

Expert Assessment. Builds on a 10-step DPM-Solver++ base and adds a critic-consistency action-refinement scheme: clipped advantage-matching actor regularisation during training and value-guided top-k action selection at inference (excerpt below shows the docstring and training setup; the full `pipeline` *function is around 320 lines). The ensemble selection idea is well-motivated within the offline-RL community, but the task explicitly asks for an inference-time-only improvement and this method also touches training behaviour, so it does not respect the spirit of the editable boundary.*

Agent: Gemini 3.1 Pro.

Step-count sweep landing on DPM-Solver++ at 3 steps.

```
discount: 0.99

# Actor
solver: ode_dpmsolver++_2M
diffusion_steps: 100
sampling_steps: 3
predict_noise: True
ema_rate: 0.995
actor_learning_rate: 0.0003
... 13 lines elided ...
ckpt: latest
num_envs: 50
num_episodes: 3
num_candidates: 100
temperature: 0.5
use_ema: True

```

Expert Assessment. Stays inside the YAML-only edit surface as intended. Tests several DPM-Solver++ configurations including a 3-step 2M-order variant, ultimately submitting that 3-step config. Effectively a hyperparameter sweep with no methodological novelty — which is the expected outcome given the narrow editable region. Useful as contrast for the GPT-5.4 attempt.

F.6. Guided Diffusion Sampling for Robot Actions

Task. An improved guidance mechanism is evaluated for a fixed trajectory-level diffusion planner on offline D4RL MuJoCo benchmarks. Editable: the network, classifier or condition module, update call, guidance weights, candidate re-ranking, and sampling logic inside `custom_guidance.py`. Read-only: the D4RL dataset and environment loop, evaluation protocol, top-level training hyperparameters, and final normalized-score aggregation. Provided baselines are Diffuser classifier guidance, a minimal classifier-free guidance ablation, `no_guidance`, and the Decision Diffuser architecture.

```
import os

import d4rl
import gym
import hydra
import numpy as np
import torch
from torch.optim.lr_scheduler import CosineAnnealingLR
from torch.utils.data import DataLoader

from cleandiffuser.classifier import CumRewClassifier
from cleandiffuser.dataset.d4rl_mujoco_dataset import D4RLMuJoCoDataset
... 3 lines elided ...
from cleandiffuser.nn_diffusion import JannerUNet1d
from cleandiffuser.utils import report_parameters
from utils import set_seed


@hydra.main(config_path="../configs/custom/mujoco", config_name="mujoco", version_base=None)
def pipeline(args):

    set_seed(args.seed)
```

```
25
26      save_path = f'results/{args.pipeline_name}/{args.task.env_name}/'
27      if os.path.exists(save_path) is False:
28          os.makedirs(save_path)
38      # ============================================================================
39      # EDITABLE REGION 3: Network + Agent Setup (lines 40-72)
40      # ============================================================================
41
42      # --------------- Network Architecture -----------------
43      nn_diffusion = JannerUNet1d(
44          obs_dim + act_dim, model_dim=args.model_dim, emb_dim=args.model_dim, dim_mult=args.task.dim_mult,
45          timestep_emb_type="positional", attention=False, kernel_size=5)
46      nn_classifier = HalfJannerUNet1d(
47          args.task.horizon, obs_dim + act_dim, out_dim=1,
48          model_dim=args.model_dim, emb_dim=args.model_dim, dim_mult=args.task.dim_mult,
49          timestep_emb_type="positional", kernel_size=3)
   ... 74 lines elided ...
124
125      # ---------------------- Inference ----------------------
126      elif args.mode == "inference":
127
128          # ============================================================================
129          # EDITABLE REGION 5: Inference Setup (lines 186-197)
130          # ============================================================================
131
132          agent.load(save_path + f"diffusion_ckpt_{args.ckpt}.pt")
133          agent.classifier.load(save_path + f"classifier_ckpt_{args.ckpt}.pt")
134
135          agent.eval()
136
145          # ============================================================================
146          # EDITABLE REGION 6: Prior + Condition Initialization (lines 207-222)
147          # ============================================================================
148
149          prior = torch.zeros((args.num_envs, args.task.horizon, obs_dim + act_dim), device=args.device)
150
151          for i in range(args.num_episodes):
152
164                  # ============================================================================
165                  # EDITABLE REGION 7: Action Sampling (lines 226-240)
166                  # ============================================================================
167
168                  # sample trajectories
169                  prior[:, 0, :obs_dim] = obs
170                  traj, log = agent.sample(
171                      prior.repeat(args.num_candidates, 1, 1),
172                      solver=args.solver,
173                      n_samples=args.num_candidates * args.num_envs,
174                      sample_steps=args.sampling_steps,
175                      use_ema=args.use_ema, w_cg=args.task.w_cg, temperature=args.temperature)
176
177                  # select the best plan
178                  logp = log["log_p"].view(args.num_candidates, args.num_envs, -1).sum(-1)
179                  idx = logp.argmax(0)
180                  act = traj.view(args.num_candidates, args.num_envs, args.task.horizon, -1)[
181                        idx, torch.arange(args.num_envs), 0, obs_dim:]
182                  act = act.clip(-1., 1.).cpu().numpy()
183
```

```
  - editable region begins at line 1 -
1 import os
2
3 import d4rl
4 import gym
5 import hydra
6 import numpy as np
7 import torch
8 from torch.optim.lr_scheduler import CosineAnnealingLR
```

```
9 from torch.utils.data import DataLoader
10
11 from cleandiffuser.classifier import CumRewClassifier
12 from cleandiffuser.dataset.d4rl_mujoco_dataset import D4RLMuJoCoDataset
13 from cleandiffuser.dataset.dataset_utils import loop_dataloader
14 from cleandiffuser.diffusion import DiscreteDiffusionSDE
15 from cleandiffuser.nn_classifier import HalfJannerUNet1d
16 from cleandiffuser.nn_diffusion import JannerUNet1d
17 from cleandiffuser.utils import report_parameters
18 from utils import set_seed
19
20
21 @hydra.main(config_path="../configs/custom/mujoco", config_name="mujoco", version_base=None)
22 def pipeline(args):
23
24     set_seed(args.seed)
25
26     save_path = f'results/{args.pipeline_name}/{args.task.env_name}/'
27     if os.path.exists(save_path) is False:
28         os.makedirs(save_path)
29
30     # ---------------------- Create Dataset ----------------------
31     env = gym.make(args.task.env_name)
32     dataset = D4RLMuJoCoDataset(
33         env.get_dataset(), horizon=args.task.horizon, terminal_penalty=args.terminal_penalty, discount=args.discount)
34     dataloader = DataLoader(
... 118 lines elided ...
153             env_eval.seed(args.seed + i * args.num_envs) if hasattr(env_eval, "seed") else None; obs, ep_reward, cum_done, t =
    env_eval.reset(), 0., 0., 0
154
155             while not np.all(cum_done) and t < 1000 + 1:
156
157                 # ==========================================================================
158                 # FIXED: Observation Normalization (lines 223-225)
159                 # ==========================================================================
160
161                 # normalize obs
162                 obs = torch.tensor(normalizer.normalize(obs), device=args.device, dtype=torch.float32)
163
164                 # ==========================================================================
165                 # EDITABLE REGION 7: Action Sampling (lines 226-240)
166                 # ==========================================================================
167
168                 # sample trajectories
169                 prior[:, 0, :obs_dim] = obs
170                 traj, log = agent.sample(
171                     prior.repeat(args.num_candidates, 1, 1),
172                     solver=args.solver,
173                     n_samples=args.num_candidates * args.num_envs,
174                     sample_steps=args.sampling_steps,
175                     use_ema=args.use_ema, w_cg=args.task.w_cg, temperature=args.temperature)
176
177                 # select the best plan
178                 logp = log["log_p"].view(args.num_candidates, args.num_envs, -1).sum(-1)
179                 idx = logp.argmax(0)
180                 act = traj.view(args.num_candidates, args.num_envs, args.task.horizon, -1)[
181                       idx, torch.arange(args.num_envs), 0, obs_dim:]
182                 act = act.clip(-1., 1.).cpu().numpy()
183
- editable region ends at line 183 -
184                 # ==========================================================================
185                 # FIXED: Environment Step + Reward Collection (lines 241-252)
```

Agent: Gemini 3.1 Pro.

Hybrid CG+CFG with normalised returns and classifier rerank.

```
14 from cleandiffuser.diffusion import DiscreteDiffusionSDE
15 from cleandiffuser.nn_classifier import HalfJannerUNet1d
```

```
from cleandiffuser.nn_diffusion import JannerUNet1d
import torch.nn as nn
from cleandiffuser.nn_condition import MLPCondition
from cleandiffuser.utils import DD_RETURN_SCALE
from cleandiffuser.utils import report_parameters
from utils import set_seed

... 27 lines elided ...
        args.task.horizon, obs_dim + act_dim, out_dim=1,
        model_dim=args.model_dim, emb_dim=args.model_dim, dim_mult=args.task.dim_mult,
        timestep_emb_type="positional", kernel_size=3)
    nn_condition = MLPCondition(
        in_dim=1, out_dim=args.model_dim,
        hidden_dims=[args.model_dim, ], act=nn.SiLU(), dropout=args.label_dropout)

    print(f"====================== Parameter Report of Diffusion Model =======================")
    report_parameters(nn_diffusion)
... 12 lines elided ...

    # --------------- Diffusion Model --------------------
    agent = DiscreteDiffusionSDE(
        nn_diffusion, nn_condition,
        fix_mask=fix_mask, loss_weight=loss_weight, classifier=classifier, ema_rate=args.ema_rate,
        device=args.device, diffusion_steps=args.diffusion_steps, predict_noise=args.predict_noise)

... 14 lines elided ...

        for batch in loop_dataloader(dataloader):

            return_scale = DD_RETURN_SCALE[args.task.env_name]
            obs = batch["obs"]["state"].to(args.device)
            act = batch["act"].to(args.device)
            val = batch["val"].to(args.device) / return_scale
... 12 lines elided ...
        condition = torch.ones((args.num_envs * args.num_candidates, 1), device=args.device) * args.task.target_return

        for i in range(args.num_episodes):

            obs, ep_reward, cum_done, t = env_eval.reset(), 0., 0., 0

            while not np.all(cum_done) and t < 1000 + 1:

... 10 lines elided ...

                # sample trajectories
                prior[:, 0, :obs_dim] = obs

                return_scale = DD_RETURN_SCALE[args.task.env_name]
                w_cg_eff = args.task.w_cg * return_scale
                w_cfg_eff = args.task.w_cfg

                # Increasing schedule: more guidance at small t (late-time steps)
                w_cfg_sched = (np.linspace(0.0, 2.0, args.sampling_steps) * w_cfg_eff).tolist()
                w_cg_sched = (np.linspace(0.0, 2.0, args.sampling_steps) * w_cg_eff).tolist()

                traj, log = agent.sample(
                    prior.repeat(args.num_candidates, 1, 1),
                    solver=args.solver,
                    n_samples=args.num_candidates * args.num_envs,
                    sample_steps=args.sampling_steps,
                    use_ema=args.use_ema,
                    condition_cfg=condition,
                    w_cfg=w_cfg_sched,
                    w_cg=w_cg_sched,
                    temperature=args.temperature)

                # select the best plan
                logp = log["log_p"].view(args.num_candidates, args.num_envs, -1).sum(-1)
```

Expert Assessment. The strongest agent submission on this task. Combines an `MLPCondition`*-based CFG path on normalised returns with the cumulative-reward classifier guidance path on unnormalised returns, then re-ranks 64 candidate trajectories by classifier log-probability. Implementation cleanly respects the editable boundary. Methodologically still a recombination of CG and CFG, but the engineering is careful and the empirical result is the best agent score on this task (hopper 0.964 / walker2d 0.785 / halfcheetah 0.442).*

Agent: GPT-5.4.

HEDGE: a hybrid that destabilises the CG baseline.

```
import hydra
import numpy as np
import torch
import torch.nn as nn
from torch.optim.lr_scheduler import CosineAnnealingLR
from torch.utils.data import DataLoader

... 2 lines elided ...
from cleandiffuser.dataset.dataset_utils import loop_dataloader
from cleandiffuser.diffusion import DiscreteDiffusionSDE
from cleandiffuser.nn_classifier import HalfJannerUNet1d
from cleandiffuser.nn_condition import MLPCondition
from cleandiffuser.nn_diffusion import JannerUNet1d
from cleandiffuser.utils import report_parameters, DD_RETURN_SCALE
from utils import set_seed


... 13 lines elided ...
    dataloader = DataLoader(
        dataset, batch_size=args.batch_size, shuffle=True, num_workers=4, pin_memory=True, drop_last=True)
    obs_dim, act_dim = dataset.o_dim, dataset.a_dim
    # ==========================================================================
    # HEDGE: Hybrid Ensemble Diffusion Guidance
    # - classifier-free return conditioning for direct target control,
    # - classifier guidance for value-gradient sharpening,
    # - stronger condition dropout to improve hybrid extrapolation.
    # ==========================================================================

    return_scale = DD_RETURN_SCALE[args.task.env_name]
    cond_dropout = max(float(getattr(args, "label_dropout", 0.25)), 0.25)

    # --------------- Network Architecture -----------------
    nn_diffusion = JannerUNet1d(
        obs_dim + act_dim, model_dim=args.model_dim, emb_dim=args.model_dim, dim_mult=args.task.dim_mult,
        timestep_emb_type="positional", attention=False, kernel_size=5)
... 92 lines elided ...
                    if count <= 0:
                        continue
                    cond_chunk = torch.clamp(rtg_now * scale, min=0.05, max=target_return_cap)
                    condition_chunks.extend([cond_chunk] * count)
                    candidate_scales.extend([scale] * count)

                num_candidates = len(candidate_scales)
                condition = torch.cat(condition_chunks, dim=0)
                traj, log = agent.sample(
                    prior.repeat(num_candidates, 1, 1),
                    solver=args.solver,
                    n_samples=num_candidates * args.num_envs,
                    sample_steps=args.sampling_steps,
                    use_ema=args.use_ema,
                    condition_cfg=condition,
                    w_cfg=hybrid_w_cfg,
                    w_cg=hybrid_w_cg,
                    temperature=args.temperature)

                traj = traj.view(num_candidates, args.num_envs, args.task.horizon, -1)
                logp = log["log_p"].view(num_candidates, args.num_envs, -1).sum(-1)
                condition_bonus = 0.05 * condition.view(num_candidates, args.num_envs)
                score = logp + condition_bonus
                topk = torch.topk(score, k=min(4, num_candidates), dim=0)
```

```
217                 idx = topk.indices[0]
218                 env_idx = torch.arange(args.num_envs, device=args.device).unsqueeze(0)
219                 elite_actions = traj[topk.indices, env_idx, 0, obs_dim:]
220                 elite_weights = torch.softmax(topk.values / 2.0, dim=0).unsqueeze(-1)
221                 act = (elite_actions * elite_weights).sum(0)
222                 act = act.clip(-1., 1.).cpu().numpy()
223                 logp = score
  - editable region ends at line 223 -
224                 # ==========================================================================
225                 # FIXED: Environment Step + Reward Collection (lines 241-252)
```

Expert Assessment. Proposes "HEDGE" — Hybrid Ensemble Diffusion Guidance — layering classifier-free return conditioning, classifier guidance, stronger condition dropout, and online remaining-return tracking at inference. The high-level idea (CFG for target-return conditioning + CG for value-gradient sharpening) is reasonable, but the implementation is heavy and ends up destabilising the CG baseline rather than improving on it. Worth showing as a failure mode of over-engineered hybrids on a saturated baseline.