The Agentic Scientist: Orchestrating Coding Agents for Automated Scientific (Co-)Discovery

Robert Tjarko Lange[1,*], **Luca Grillotti**[1,*], **Yutaro Yamada**[1,*], **Rujikorn Charakorn**[1] **and David Ha**[1]
[1]Sakana AI, [*]Equal Contribution.

arXiv:2610.18933v1 [cs.AI] 13 Oct 2026

Modern coding agents are fundamentally reshaping the role of human input for computational research: they write code, launch in silico experiments, analyze their results, and autonomously operate with ever-increasing time horizons. Despite these developments, fully autonomous research agents suffer from a limited 'ideation radius' and myopic hill-climbing search dynamics. To bridge this gap, we introduce The Agentic Scientist, an end-to-end distributed pipeline for human researchers to conduct computational research alongside modern coding agents. Building on its predecessor (The AI Scientist, Lu et al., 2026), our system decomposes research into task drafting, idea generation, planning, sandboxed experimentation, result aggregation, baseline and ablation construction, post-hoc analysis, and report writing. Importantly, each step in the pipeline is powered by the next generation of headless coding agents and leverages agentic self-verification. The overall research process is driven by a meta-controller that manages parallel research agents and enforces rubric verification at each stage. This design enables the same orchestration layer to be reused across heterogeneous domains with different execution environments, agentic coding harnesses, and evaluation metrics. Our system can be used across a spectrum of autonomy: either as a co-scientist tool, or to perform end-to-end scientific investigation. Within the co-scientist setting, we leveraged a development version of our system to submit a single research paper, which got accepted for publication at ALIFE 2026 (Yamada et al., 2026). We further apply The Agentic Scientist fully autonomously to open problem-solving tasks in machine learning, and demonstrate the system's capability of scientific discovery across different domains of computational research, including: a hypernetwork-based inductive bias for rapid Transformer adaptation and a meta-trained evolution strategy that matches or exceeds CMA-ES on 5 of 7 unseen Brax control tasks. Our ablations suggest that reliable scientific autonomy depends not only on base model capabilities, but also on the cross-agent orchestration layer. We open-source The Agentic Scientist and its figure generation tool, autoillustrate, to aid collective development of automated (co-)scientist systems:

Agentic Scientist https://github.com/SakanaAI/AI-Scientist-v3
autoillustrate https://github.com/SakanaAI/Autoillustrate

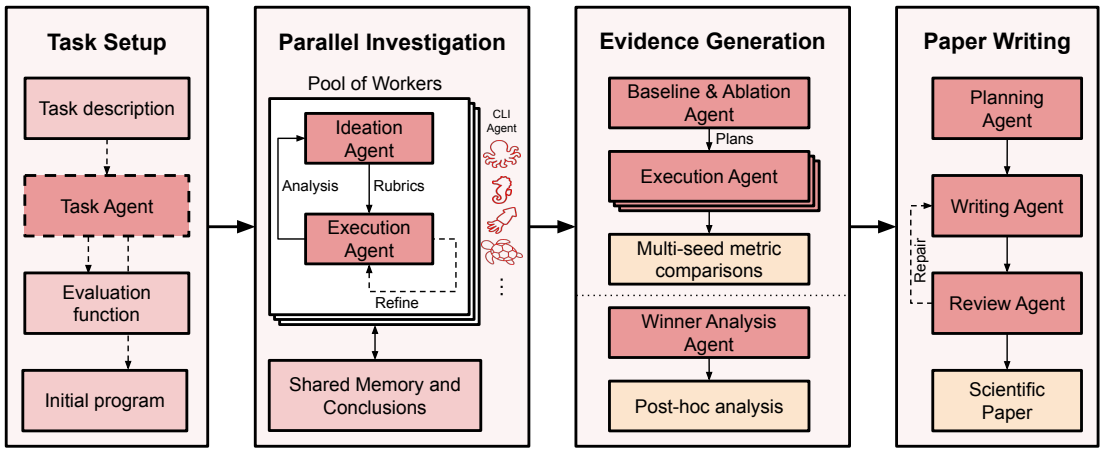


Figure 1 | The Agentic Scientist Four Stage Workflow.

Corresponding author(s): Robert Tjarko Lange (robert@sakana.ai). Full author contributions list in Section 8.

1. Introduction

Large Language Model (LLM)-based tool-calling agents can now complete increasingly long coding tasks with minimal human intervention (Kwa et al., 2025). Computational research, however, is arguably more than a single long coding task: a practical system supporting human scientists must generate hypotheses, execute controlled experiments, validate claims, iterate on ideas, and communicate the results. Existing AI scientist systems demonstrate parts of this loop (Jansen et al., 2025; Karpathy, 2026; Lu et al., 2024b; Schmidgall et al., 2025; Tang et al., 2025; Yamada et al., 2025), but myopic hill-climbing can make agents behave merely as flexible hyperparameter optimization engines (Ferreira et al., 2026; Van Stein et al., 2024).

Here, we tackle the systems-level gap between an agent that writes code and a system that can conduct reliable research. We introduce The Agentic Scientist (Figure 1), a human-steerable, agentic scaffold for end-to-end computational research. The workflow drafts a task specification from a text description, coordinates parallel sandboxed workers, accumulates retrospective evidence, and writes a manuscript grounded in the results. During the investigation phase, parallel workers revise scientific hypotheses in response to implementation failures, measured outcomes, and shared performance notes. The resulting artifacts seed subsequent experiments and remain available for retrospective analysis, ablation design, and manuscript writing. Predecessors of our system (Table 1) relied on initial implementation directions and loose assessment of the idea implementation. This could result in weak experiment validation and overfitting. The Agentic Scientist, on the other hand, uses self-drafted rubrics-based verification and CLI agents for broader code edits and information protocols.

Table 1 | **Comparison of Sakana AI's AI Scientist Frameworks.** The Agentic Scientist enhances previous versions with rubric-based verification and the option for co-scientist enhancement.

Feature	**Codebase Drafting**	**Execution Workflow**	**Parallel Experiments**	**Rubric-based Self-Verification**	**Human Evaluation Result**
AI Scientist-v1	Topic-Specific	Linear (Aider)	✗	✗	Not Submitted
AI Scientist-v2	Domain-General	Tree-Based (AIDE)	✓	✗	Conference Workshop Acceptance
Agentic Scientist	Domain-General	Beam Search (CLI Agents)	✓	✓	Co-Scientist Conference Acceptance

A development version of our system was used to automate parts of the research process on Error Diffusion (Fujita, 2026; Kaneko, 2000). Afterwards, we submitted a single paper which was accepted as an ALIFE 2026 talk (Yamada et al., 2026). We further demonstrate our framework across three Machine Learning research directions: our system discovers composition and optimization biases for hypernetwork-based LLM adaptation (Charakorn et al., 2026). The Agentic Scientist also extends research on neural network-parametrized meta-black-box optimization (Ma et al., 2025), yielding an optimizer now integrated in `evosax` (Lange, 2023). Finally, our system manages to iteratively improve an RL-based LLM fine-tuning program by combining recent algorithmic advances. All of these demonstrate the broad application of the system to enhance scientific knowledge beyond mere hill-climbing applications. Our main contributions are:

1. **The Agentic Scientist**, a (co-)scientist orchestration scaffold for human-steerable ideation, parallel execution, shared memory, rubric-based verification, and paper manuscript writing.
2. **Co-scientist paper conference acceptance.** A paper created with assistance by The Agentic Scientist on backpropagation-free error diffusion learning algorithms was accepted at ALIFE 2026 (Yamada et al., 2026) after limited human edits and verification.
3. **Cross-domain & harness validation** spanning long-context LLM adaptation, evolutionary optimization, RL fine-tuning, and 5 additional domains. Across various tasks, model harnesses, and base models, our system managed to generate high-quality grounded research manuscripts.
4. **autoillustrate**, an agentic pipeline for editable scientific figure generation that outperforms PaperBanana on PaperBananaBench (37.5 vs. 25.0 overall score, Zhu et al., 2026a).

2. Background & Related Work

End-to-end automated scientific discovery. Research on LLM-driven science increasingly targets complete workflows rather than isolated steps (Si et al., 2025). The AI Scientist-v1 introduced a closed loop for proposing hypotheses, running experiments, analyzing results, and drafting manuscripts (Lu et al., 2024b), while The AI Scientist-v2 removed human-provided code templates and introduced progressive agentic tree search (Yamada et al., 2025). Ultimately, these developments combined with base LLM capabilities have led to end-to-end autonomous systems capable of producing workshop-level research content (AutoScience AI, 2025; Intology AI, 2025; Lu et al., 2026). Agent Laboratory and AI-Researcher similarly connect literature analysis, experimentation, and report writing, with different opportunities for human feedback (Schmidgall et al., 2025; Tang et al., 2025). CodeScientist combines literature and code search with code review, replication, and external assessment (Jansen et al., 2025), while CycleResearcher inserts automated review into the loop (Weng et al., 2025) and the AI co-scientist emphasizes scientist-guided hypothesis development (Gottweis et al., 2026). These systems vary in autonomy, search, and validation; here our focus is the reusable plus scalable orchestration and evidence layer that supports heterogeneous coding agents.

Hill-Climbing via Program Evolution. A complementary line casts discovery as search over executable artifacts. FunSearch showed that language-model-guided program search can yield mathematical discoveries (Romera-Paredes et al., 2024); AlphaEvolve broadened evolutionary code search to algorithms and computational infrastructure (Novikov et al., 2025), and ShinkaEvolve targets open-ended, sample-efficient program evolution (Lange et al., 2026). At a lighter weight, `autoresearch` uses coding agents to repeatedly edit and evaluate a training program under a fixed budget (Karpathy, 2026), motivating direct comparison with classical hyperparameter optimization (Ferreira et al., 2026). ALE-Agent (Imajuku et al., 2025) demonstrated that simple beam search at sufficient scale provides a competitive harness for program optimization. Across these approaches, the search unit is a program under a pre-specified objective. Our scaffold uses such executable search within a broader hypothesis-to-report process: it broadens the scope of application by automating task drafting, validation rubric generation, and supporting objective creation.

Benchmarking Research Agent Capabilities. Research-agent benchmarks separate several capabilities that a single final score can obscure. SWE-agent (Yang et al., 2024b) established that the agent–computer interface strongly affects real-world coding performance. MLAgentBench evaluates iterative machine-learning experimentation (Huang et al., 2024), whereas MLE-Bench covers longer-horizon engineering workflows involving data preparation, debugging, training, and optimization (Chan et al., 2025); subsequent work studies how search strategy affects exploration and generalization in this setting (Toledo et al., 2025). At the research level, PaperBench measures replication of published AI work (Starace et al., 2025), MLR-Bench targets open-ended machine-learning research (Chen et al., 2025), and EXP-Bench isolates whether agents can conduct valid AI experiments (Kon et al., 2026). Lu et al. (2024b) introduced an LLM-driven pipeline for paper reviewing, which was later validated on research papers published after the training data cutoff date (Lu et al., 2026). Here, we make use of such automated proxy AI reviews and additionally report external reviews of the co-produced ALIFE submission as complementary evidence.

Verification and Grounding. Verification and communication are becoming explicit parts of agent design. Rubric rewards can train AI co-scientists and provide feedback in domains without verifiable objectives (Goel et al., 2026; Gunjal et al., 2026). Here, rubrics do not train the coding-agent backbone; instead, they serve as runtime contracts that ground each hypothesis to implementation checks and empirical evidence. The The Agentic Scientist manuscript drafting runs additional checks for facts and figures outlined in the paper. We discuss further related work in Section A.

3. The Agentic Scientist: An Orchestration Layer for Research Agents

The Agentic Scientist is a versatile agentic orchestrator scaffold for end-to-end computational *research campaigns* (Figure 1). A campaign begins with a task specification, which researchers can provide directly or generate from a research brief with a drafting agent. Then, the campaign proceeds through three phases: parallel investigation, retrospective evidence generation, and grounded paper writing. Each phase runs as a separate process and feeds the next: the selected program and experimental artifacts produced by the first phase determine which baselines, ablations, and post-hoc analyses to run; the resulting evidence then grounds the manuscript. Finally, we leverage the previously introduced AI reviewer (Lu et al., 2026) to provide proxy evaluation scores for the manuscripts.

3.1. Research Task Setup

Before idea execution, The Agentic Scientist requires an executable task specification that defines the research goal and its evaluation. This can either be specified manually or drafted agentically:

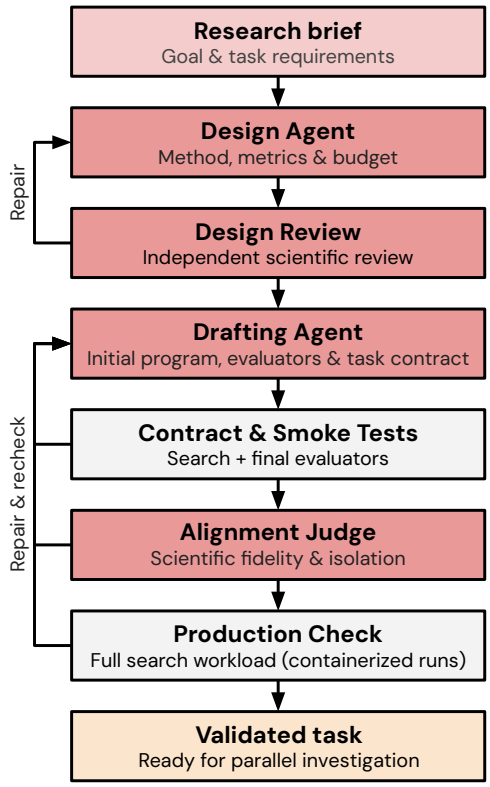


Figure 2 | Task drafting workflow.

Task specification. A task specification comprises an initial program, a search evaluation program, an optional final evaluation program, and a machine-readable contract: the initial program provides a deterministic baseline, while the search evaluation program determines whether a candidate program is correct, returns the scalar score used to rank and select candidates during the investigation phase, and records metrics and artifacts for later analysis. When provided, the final evaluation program runs the selected program after exploration and reports post-search evaluation metrics. The contract defines the candidate interface and evaluation procedures. Tasks may additionally stage papers or code as research context, as in our hypernetwork adaptation and error-diffusion studies (Sections 4 and 6.1).

Task drafting. Researchers may provide the task specification directly or use the task drafting pipeline to create it from a human-written research brief (Figure 2). The pipeline has two main stages: a design agent converts the research brief into an experimental design, and a drafting agent converts this design into the task specification.

The experimental design defines the method, candidate interface, compute budget, and protocols for search and final evaluations. Before drafting begins, an independent review agent assesses the design and repairs any issues. The drafting agent then implements the initial program, the search and final evaluation programs, and the task contract, while pinning dependencies and staging the required data. The harness checks the contract and runs smoke tests of the evaluation programs before the alignment judge assesses scientific fidelity and verifies that the search and final evaluators are independent and use disjoint data. In containerized runs, the production check additionally requires the initial program to complete the full search workload to ensure operational feasibility. Failed scaffolds are repaired, and only versions that pass all checks proceed to parallel investigation.

3.2. Parallel Investigation

The idea investigation phase spends a fixed budget of containerized research cycles across parallel workers, each exploring different hypotheses in a separate workspace while sharing distilled knowledge (Figure 3). Multiple agent-based workers explore ideas in parallel, each advancing an independent sequence of research cycles. These workers lease cycles from a synchronized queue, maintain separate run directories and worker-local best search score programs, which resemble a per-worker beam search. A cycle starts with the worker importing the shared memory and the other workers' distilled conclusions, and ends with the worker publishing new memory entries and updated conclusions to all other workers. Communication occurs at cycle boundaries, while logs and program artifacts remain in worker directories. At regular intervals, optional context pruning archives the histories and memory before resetting them, so that accumulated context does not bias subsequent cycles; the best program is carried into the next round to preserve progress. After parallel execution, the aggregator merges worker histories and selects the best correct program found across workers. This selected program becomes the input to the retrospective evidence-generation phase.

3.3. Contracted Research Cycles

Each worker advances a sequence of research cycles, and each cycle runs two agents in turn: an ideation agent and an execution agent. The ideation agent turns one hypothesis into a typed contract, and the execution agent then tests that contract against the evaluator, so that intent and evidence remain separate(Figure 3).

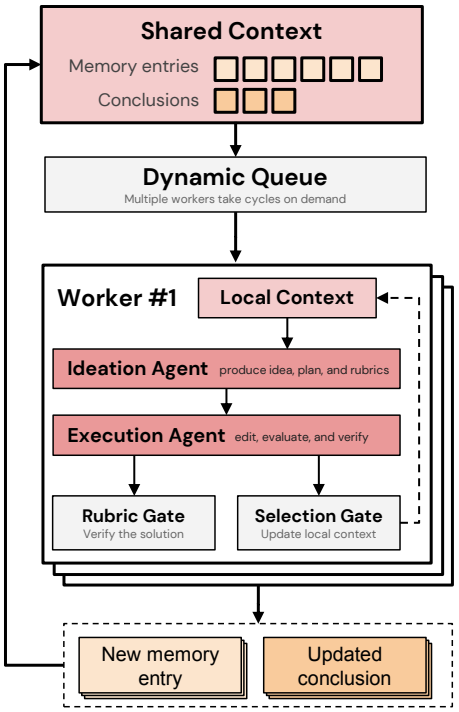


Figure 3 | Investigation workflow.

Ideation agent. At the start of a cycle, the ideation agent receives the task, recent idea and attempt histories, the manifest for the worker's best program, and distilled conclusions. The protocol requires inspecting this context without experimenting or modifying candidates ('read-only'). The ideation agent must instead emit a structured idea bundle containing a title, a falsifiable hypothesis, an ordered implementation plan, and the target rubrics. Those records are normalized, and any missing fields or mismatched rubric identifiers are rejected. This boundary records the idea as a structured artifact, so the execution agent does not have to reconstruct its intent from conversational context.

Execution agent. The accepted contract, evaluator, task files, and the worker's best program are staged into an experiment workspace. The execution agent creates append-only candidate variants and follows an edit–evaluate–verify loop within a fixed evaluation budget, producing metrics, correctness checks, and rubric evidence. Two distinct gates are then applied: an all-rubric-pass attempt marks the contract complete, whereas updating the worker's best program requires a correct candidate with a strictly higher score; rubric completion and updates to the worker's best program are therefore independent decisions. All logged idea provenance and attempt outcomes are retained and later condensed to guide interpretation during paper writing.

3.4. Retrospective Evidence Generation

This phase studies the selected program from the investigation phase in depth: (1) it runs controlled baselines and ablations that isolate the idea effects, and (2) it derives post-hoc analyses that characterize the behavior of the method on the considered task.

Controlled baselines and ablations. The selected program is first instrumented with additional metrics, such as extra scores and time series data, so that every generated baseline and ablation can be compared in detail on common ground. Given the code diff between the selected and initial programs, an agent then proposes baselines and ablations with explicit hypotheses and implementation plans, where each ablation must isolate an innovation introduced by the selected program. A separate agent filters these proposals for relevance, feasibility, and single-change scope. Each validated plan receives an isolated workspace and follows an agent-guided edit–evaluate–verify loop that modifies the selected program to implement the intended change and score it against the evaluator, alongside the unchanged selected and initial programs. The evaluator can be run for multiple seeds to make comparisons statistically meaningful, and their summaries and metrics are aggregated across seeds, ablations, and baselines by task.

Post-hoc analysis. The post-hoc pipeline proposes, filters, executes, and aggregates diagnostics that characterize the behavior of the frozen selected program on the considered task. Diagnostics may run new inference or analyze existing artifacts, such as checkpoints and logs. Programmatic checks reject proposals that retrain or otherwise alter the selected program, or require unavailable artifacts. When no usable checkpoint exists, an optional bootstrap stage enables checkpoint saving and reruns the evaluator once; all subsequent analyses keep the resulting version frozen. The resulting post-hoc analyses, together with the baseline and ablation results, form the evidence that the paper writing phase turns into empirical claims.

3.5. Grounded Paper Writing

The paper writing phase converts the outputs of the preceding phases into a manuscript through an artifact-grounded workflow that reviews and audits claims made in the paper (Figure 4).

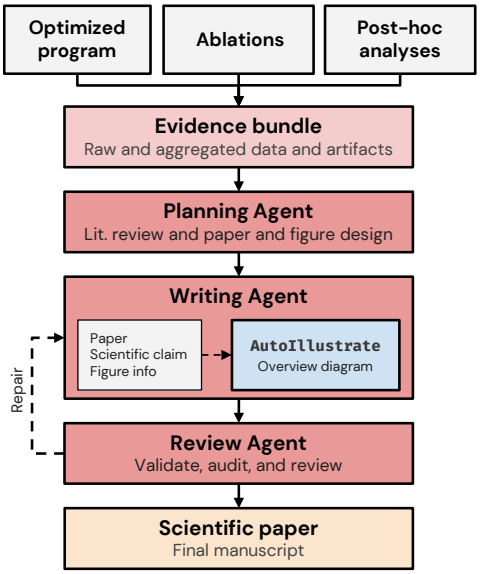


Figure 4 | Evidence-to-paper pipeline.

Evidence staging and planning. We merge worker results into aggregate outputs and stage them in a dedicated workspace, with a memory summary and raw artifacts. Quantitative claims resolve to those outputs and raw artifacts, while the memory summary only guides interpretation. Before and after drafting, reference planning retrieves bibliography entries (Section F.1). An optional stage fixes the central thesis, result emphasis, and figure plan.

Manuscript generation. The writer agent produces a compilable LaTeX manuscript together with a claim-to-source map and figure manifest, whose plots are derived from staged workspace results. The strategy may also request AUTOILLUSTRATE, our agentic figure generation pipeline (Section 3.6), to produce a conceptual overview diagram.

Validation and review. At the end of our pipeline, a single repair loop alternates deterministic checks with agent judgment and feeds any failures back to the writer agent for revision. The validation checks cover structural completeness, provenance and claim-to-source links, compilation, and venue and page budget constraints. An independent audit agent runs every round on a read-only copy of the workspace and flags where the paper's numbers, comparisons, or conclusions overclaim or contradict the canonical data and evidence trail. An optional reviewer-style agent assesses framing, caveats, figures, and clarity, and may ask the writer agent to narrow or qualify claims.

3.6. AUTOILLUSTRATE: Conceptual Diagram Generation

AUTOILLUSTRATE is designed around two requirements: scientists should be able to modify conceptual figures, and they should represent the methodology faithfully. To meet these requirements, the pipeline first produces an editable `draw.io` diagram to ground the figure's structure, which is later exported and augmented using a separate image generation model. As illustrated in Figure 5, the concept generator runs a six-phase pipeline that turns a method brief into a multi-panel figure whose layout source remains explicitly editable. Unless stated otherwise, all papers generated by our system in this work have their main conceptual figure generated by AUTOILLUSTRATE.

Editable panel generation. The pipeline begins with a planning agent that reads the staged source and instructions and describes the panels within one top-level configurable layout. A styling agent then defines a shared color palette and typography specification to keep independently rendered panels consistent. Rendering agents create the panels independently and in parallel as editable `draw.io` diagrams. Each agent relies on a template to define the panel topology and a Python generator script to compute coordinates and check bounding boxes for overlap. Visuals that `draw.io` cannot produce remain as placeholder boxes. When possible, a generated `matplotlib` script fills these boxes; the enhancement stage handles any remaining placeholders. A vision-model audit then evaluates each panel for rendering fidelity, factual accuracy, and sufficient detail. When the inspection detects a defect, the pipeline edits the source, rebuilds the panel, and repeats the audit.

Composition, enhancement, and verification. After all panels pass, the pipeline composes them into a single editable `draw.io` source and exports a raster version. An image generation model restyles the raster export, after which a final verification step compares the enhanced candidate with the editable reference and repeats the checks defined during panel inspection. Any discrepancy triggers another source-editing, export, enhancement, and verification pass. After several repair passes, the pipeline returns the enhanced candidate with the strongest verification results. If no candidate passes verification, the pipeline returns the raster export from the editable source. Because the pipeline preserves this source throughout the process, scientists can revise the content or layout directly without regenerating the figure. We provide additional details in Section E.

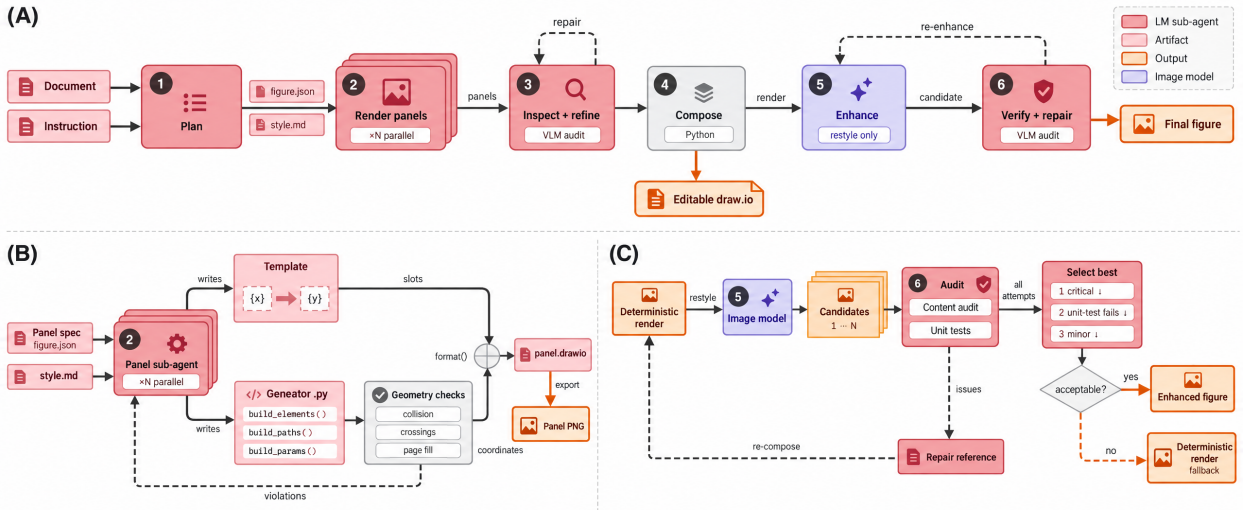


Figure 5 | **AUTOILLUSTRATE pipeline**, generated by AUTOILLUSTRATE itself. **(A)** Six phases turn a document and an instruction into an editable `draw.io` figure: LLM sub-agents decide content, while deterministic Python owns geometry and the generative image model only restyles. **(B)** In Phase 2, panels are rendered from a combination of automatically-generated templates whose coordinates are determined by LLM-generated Python code. **(C)** In Phases 5–6, the enhanced candidate with the best audit results is selected; if none is acceptable, the deterministic render is used as a fallback.

4. AI Co-Scientist and Conference Paper Submission

We validated a development version of our system within a co-scientist setting: The Agentic Scientist was used to automate parts of the research process targeting the extension of a backpropagation-free neural network training method. After thorough human verification and manuscript edits, we submitted a single paper to ALIFE 2026, which was accepted after external peer review as a main conference talk. We emphasize that this single submitted ALIFE paper was produced by human researchers using The Agentic Scientist as a tool, and not submitted as an unaltered system output. The human authors remain responsible for its methods, claims, authorship, and disclosure. In this section, our focus is to describe the process by which experimental outputs became a submitted manuscript and were subsequently revised for the camera-ready version (Figure 6).



Figure 6 | Camera-ready ALIFE 2026 paper after the single submitted manuscript was accepted. The full manuscript and additional details can be found in Section D.

4.1. A Human-Supervised Research Workflow

The project began with a human-selected research direction: investigating how Error Diffusion (ED) could be extended beyond its earlier demonstrations (Fujita, 2026; Kaneko, 2000). ED leverages a three-factor learning rule (presynaptic activity, postsynaptic activity, and a globally broadcast error signal) originally studied in a dual-stream setting with paired excitatory (E) and inhibitory (I) neurons. We chose to illuminate this research direction together with The Agentic Scientist since it investigates the following question: how can classical biologically-inspired learning algorithms be revived with modern deep learning techniques?

Human researchers specified the scientific objective and the constraints for a valid implementation. Within that contract, The Agentic Scientist proposed candidates, implemented training procedures, ran experiments, inspected failures, and generated plots. The human researchers selected benchmarks (see creation process overview in Figure 7) and specified initial rubrics: A numerically

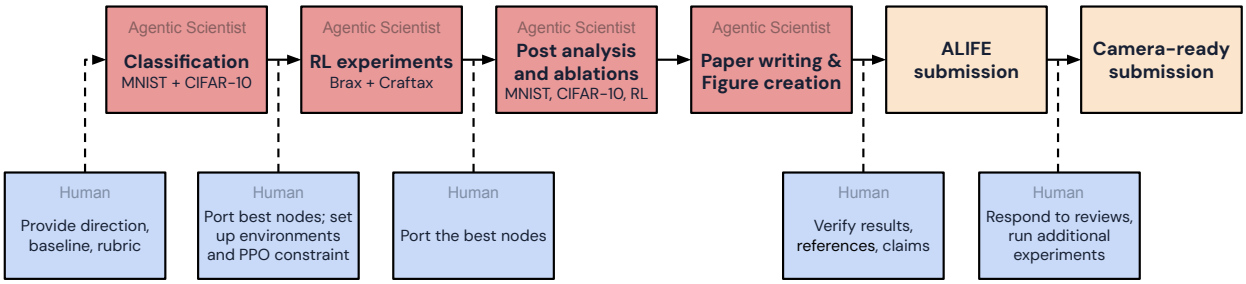


Figure 7 | Creation process of the ALIFE paper: human researchers defined the initial direction, baseline, and rubrics; supported the experiment domain transition; verified results, references, and claims before submission; and incorporated reviewer feedback into the camera-ready paper.

strong result was not treated as evidence of progress unless its implementation also satisfied the rubric constraints. For example, one rubric prohibited hidden credit assignment through automatic differentiation or transposed forward weights. This required the positive and negative streams to remain distinct dual-stream, and specified the intended local information available to an update.

4.2. Summary of Discovery and Dynamics

The Agentic Scientist first extended ED from binary to multiclass classification: during the parallel investigation phase, The Agentic Scientist developed *modulo error routing*, a structured mechanism for assigning multi-class output errors to hidden units. This enabled multiclass learning while preserving the dual-stream, Dale-constrained architecture, in which weight signs remain fixed throughout training. Subsequent experiments focused on robustifying learning dynamics: The Agentic Scientist identified layer-specific sigmoid widths, batch-centered error signals, and asymmetric E/I initialization as important stabilizers, raising accuracy to 96.7% on MNIST (LeCun et al., 1998) and 61.7% on CIFAR-10 (Krizhevsky et al., 2009). Ablations further showed that the individual contributions were task-dependent: widened sigmoids were critical on MNIST, whereas batch-centering became substantially more important on CIFAR-10.

At this stage, human researchers transferred the best-performing ED implementation from the MNIST and CIFAR-10 experiments to the RL setting, and prepared the Brax (Freeman et al., 2021) and Craftax (Matthews et al., 2024) environments. For RL experiments, an additional methodological constraint was added to the rubrics: the method was required to retain the PPO objective (Schulman et al., 2017), while using the gradient of the PPO loss with respect to the logits as the output-level error signal for Error Diffusion. Within this constraint, The Agentic Scientist adapted the dual-stream ED mechanism to PPO and explored the resulting ED-PPO variants. ED-PPO achieved performance comparable to standard baselines on several Brax tasks, although its performance was less consistent on Humanoid and Craftax (see figures in ALIFE paper manuscript; Section C.3).

Post-hoc analyses linked these results to the dynamics of credit assignment: local update signals weakened substantially in earlier layers, while training shifted E/I ratios toward balance. In summary, the results obtained demonstrate that ED generalizes beyond its original binary-classification setting. The mechanisms required for effective learning depend on task difficulty and domain.

4.3. Human Result Verification & Manuscript Edits

Before submission to ALIFE, researchers verified the generated research artifacts. This included checking that the generated implementations matched the scientific hypotheses being tested, confirming that the reported tables and figures accurately reflected results produced by the corresponding code, and revising the manuscript where the generated text overstated or inaccurately characterized the

Table 2 | Revision statistics across manuscript stages. Counts are word-equivalents computed directly from the `latexdiff` LaTeX sources. Percentages are relative to the pre-revision manuscript length.

Revision stage	Inserted	Deleted	Net change
Agentic Scientist → submitted	664 (17.5%)	405 (10.7%)	+259 (+6.8%)
Submitted → camera-ready	1,329 (32.8%)	630 (15.6%)	+699 (+17.3%)

evidence. For example, human researchers added appropriate statistical analyses and, when the results showed comparable performance, revised claims such as "ED-PPO significantly exceeds BP-PPO" to "ED-PPO performs on par with BP-PPO". Additionally, all references in the BibTeX bibliography were manually verified to ensure that the cited works existed and supported the statements for which they were cited. During this process, human researchers removed several citations that did not support the corresponding claims, and added important references omitted from the generated manuscript. These included the reference to the original ED work, which is available only through an archived webpage. The generated manuscript was treated as an intermediate research artifact, which was modified by human researchers, improving the writing of several sections (e.g., future work, etc.). We then submitted this single paper to the ALIFE 2026 conference.

4.4. Peer Review, Camera-Ready Paper Revisions

This submission process illustrates the AI co-scientist approach enabled by our system: outputs were not just optimized code or isolated experimental reports, but a coherent working paper making a scientific claim, positioning it relative to prior work, and presenting evidence across multiple domains. We emphasize that only a single paper was submitted after our careful manual verification.

The subsequent ALIFE peer review process provides external evidence for the quality of the co-created paper submission. Overall, the paper was accepted as a talk, with individual recommendations of *Strong ACCEPT*, *Weak REJECT*, and *Weak ACCEPT*:

`Reviewer 1` framed the work as a relevant artificial-life contribution, as it evaluates the consequences of a biological assumption, Dale's principle, and found the methods and results technically sound.
`Reviewer 2` highlighted that the neuroscience connection of the work was interesting but questioned whether the paper would better fit a neural network venue. Additionally, they asked for clearer explanations of notation, matrix conventions, modulo indexing, and technical details.
`Reviewer 3` was positive and requested more intuition about ED updates, non-negative activations, per-neuron error signals, baseline comparisons, ED-PPO architecture, and the meaning of gradient attenuation without backpropagation.

We afterward used this feedback to enhance the camera-ready version of the manuscript: the post-acceptance changes are best characterized as targeted scientific clarifications rather than a change in the underlying study. In particular, the camera-ready version tightens several claims about prior ED results, adds Dale-compliant reinforcement-learning results, corrects the Error Diffusion citation context, clarifies that Direct Feedback Alignment (Nøkland, 2016) routes errors through fixed random matrices, and cleans up the dual-stream architecture description and ED update equation. It also replaces potentially misleading gradient language with "proxy" or "local surrogate" gradient terminology, refines the biological interpretation of E/I balance, repositions the discussion of Craftax and the dual-stream parameter overhead, and fixes typographic details such as quotation marks. These edits make the paper more precise about biological plausibility and local learning while leaving the empirical evidence and overall narrative intact. Table 2 summarizes human intervention throughout the manuscript-development process. In summary, this study provides evidence of The Agentic Scientist being capable of supporting researchers to explore unorthodox AI topics across the entire research process.

5. Evaluating The Agentic Scientist Across Tasks, Harnesses, and LLMs

We evaluate The Agentic Scientist across five manually prepared research tasks that span optimization, representation learning, and predictive modelling. Each campaign starts from a task specification with different research objectives, starting points, and search iteration resources (see Table 3 for the task suite overview).

Task, harness, and model sweep. We run the task sweep across four agent harnesses: Codex, Claude Code, Pi, and OpenCode (Anomaly Innovations, 2026; Anthropic, 2026; Earendil Works, 2026; OpenAI, 2026). Codex and Claude Code each use four assigned models, while Pi and OpenCode each use three, all at high reasoning effort (Table 10). Each run uses 20 idea investigation iterations and concludes with a written manuscript. Here, we focus on aggregated insights and provide more details on task-specific settings and discoveries can be found in Section B.1.

Table 3 | Research tasks, starting implementations, and objectives used in the aggregate comparison.

Task	Starting Point	Research Objective
Learned optimization	Baseline: ELO's expert-supervised MLP optimizer, meta-trained on (Fashion-)MNIST (Huang et al., 2026b); Evaluation: CIFAR-10 and SVHN.	Improve transfer to unseen datasets and network architectures under a fixed meta-training budget. Search iteration budget: 2 H100s, 150 min.
World models	Baseline: Action-free Pong model, with a trained discrete tokenizer and masked-token dynamics; Evaluation: Error reduction against TinyWorlds	Reduce held-out free-running rollout error across prediction horizons relative to Majmudar (2026). Search iteration budget: 4 H100s, 180 min.
Sparse autoencoders (SAE)	Baseline: TopK SAE activation-trained on Pythia-160M-deduped & Gemma-2-2B-it; Evaluation: Task-specific SAEBench (Karvonen et al., 2025).	Improve reconstruction at fixed sparsity while preserving model behavior and feature utilization. Search iteration budget: 2 H100s, 60 min.
Tabular distribution shift	Baseline: TabICLv2's (Qu et al., 2026b) stage-one synthetic-prior pretraining from scratch recipe; Evaluation: 4 held-out real datasets vs. reference.	Improve in-context balanced accuracy on held-out real table data & across context sizes. Search iteration budget: 4 H100s, 150 min.
Hyperparameter optimization	Baseline: Adaptive incumbent-centred local search; Evaluation: BBOB (Hansen et al., 2021), HPO-B (Arango et al., 2021), YAHPO (Pfisterer et al., 2022).	Improve sample-efficient optimization over continuous, mixed, and hierarchical search spaces. Search iteration budget: 8 CPU cores, 30 min.

Paper-level evaluation protocol. All generated research runs share the same scoring function within a task instance and can thereby be directly compared on their numerical optimization outcome. Additionally, we use the AI Reviewer's (Lu et al., 2026) 1–10 *Overall* score as a paper-level proxy metric: For each generated manuscript, five separate calls to `o4-mini-2025-04-16` review the extracted PDF text using NeurIPS-style guidelines and a fixed in-context example of a paper–review pair. A sixth call acts as an area chair to synthesize the reviews. The original study in Lu et al. (2026) validated the reviewer against acceptance decisions for ICLR submissions from OpenReview, including a post-training-data-cutoff 2025 cohort. We note that the AI reviewer score solely serves as a proxy for reviewer-perceived paper quality, and not as ground-truth evidence of full scientific correctness.

Model capability, harness impact, and paper quality. Generally speaking, newer model releases tend to be associated with higher reviewer scores (Figure 8a). When evaluating the models within their respective native coding harnesses, this observation appears to be more strongly pronounced for Claude- than GPT-based models. Matched-model comparisons show minor performance differences across harness choice, while the underlying model matters more (Figure 8b). No harness is uniformly best. OpenCode ranks lowest with GPT-5.6-Sol but highest with Opus 5 and DeepSeek-v4-Flash. The model developers' own harnesses offer no consistent advantage: Codex ties Pi with GPT-5.6-Sol, and Claude Code falls between OpenCode and Pi with Opus 5.

Search progress and rubric dynamics. The search traces show gains in best observed scores alongside fluctuating rubric pass rates. Final performance is positively associated with reviewer

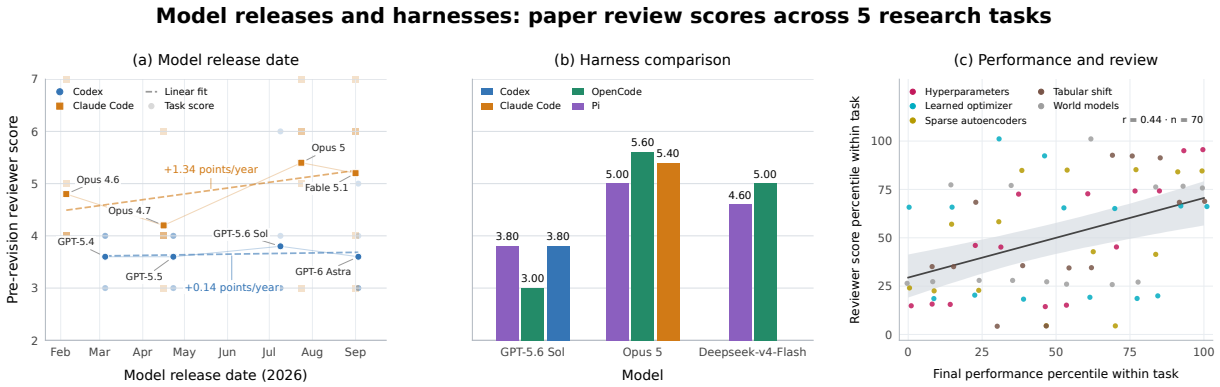


Figure 8 | Pre-revision reviewer scores by (a) model release date and (b) harness, averaged across five tasks; (c) within-task percentile ranks of final performance and reviewer scores, colored by task. The least-squares fit uses a pointwise 95% band from 10,000 bootstrap resamples of the 14 complete harness–model profiles across all five tasks, conditional on the observed tasks and fixed ranks.

scores on four of the five tasks, most strongly for hyperparameter optimization, while learned optimization shows little association (Figure 9). Pooling all 70 campaigns after converting both final performance and reviewer scores to within-task percentile ranks gives a positive aggregated association (r = 0.44; Figure 8c). This observation is consistent with manuscript reviewer scores tracking measured numerical performance.

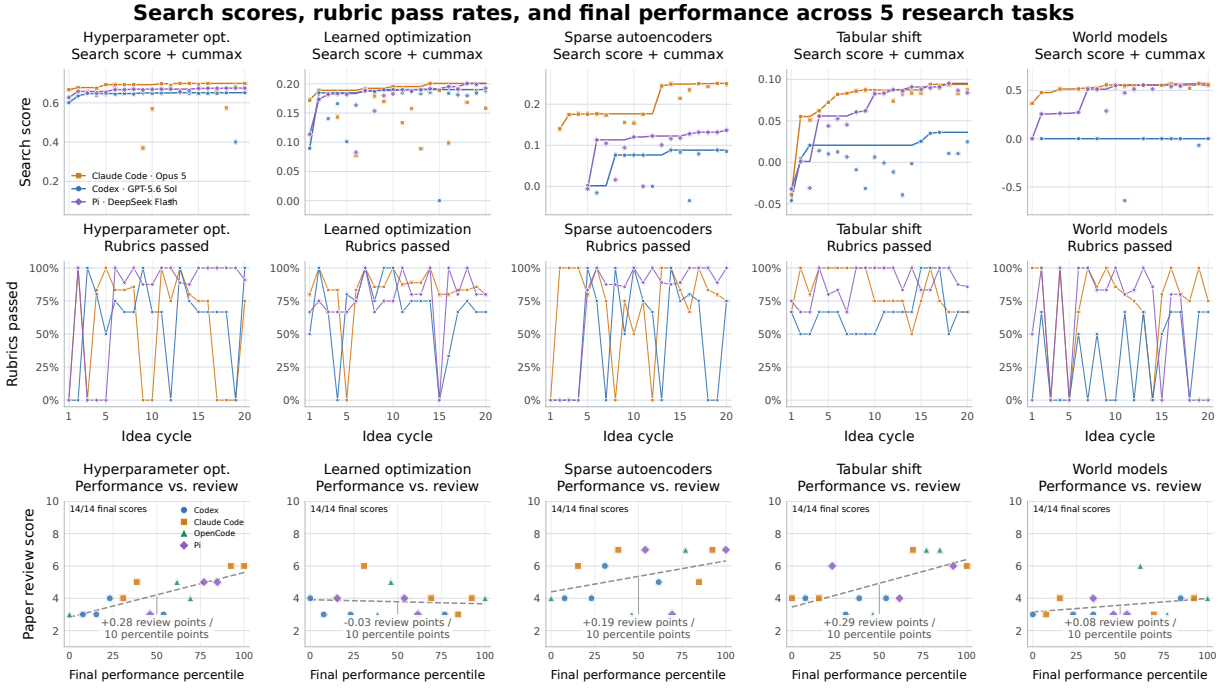


Figure 9 | Search scores (top), rubric pass rates (middle), and final performance versus reviewer score (bottom) across five tasks. Bottom panels show within-task performance percentiles against original 1–10 reviewer scores; annotated least-squares slopes give reviewer-score points per 10 percentile points. Final performance is positively associated with reviewer scores on four of the five tasks.

6. Additional Case Studies: Putting The Agentic Scientist to work

We further evaluate The Agentic Scientist on three case studies: Language model adaptation (Section 6.1) through hypernetworks (Charakorn et al., 2026), evolutionary optimization (Section 6.2) through meta-learned neural network parametrizations (Lange et al., 2023a), and language model RL fine-tuning (Section 6.3) on mathematical reasoning tasks.

6.1. Open-Ended Architecture Research with The Agentic Scientist: Discovering Query-Conditioned Composition for Hypernetwork-Based LLM Adaptation

In this experiment, we explore how The Agentic Scientist can autonomously write an insightful scientific paper after successfully investigating a specific topic. As a case study, we choose the recently published hypernetwork-based LLM adaptation work from Charakorn et al. (2026), *Doc-to-LoRA*. Unlike the other settings, The Agentic Scientist is given the Doc-to-LoRA paper's LaTeX source and its public code repository as the starting point.

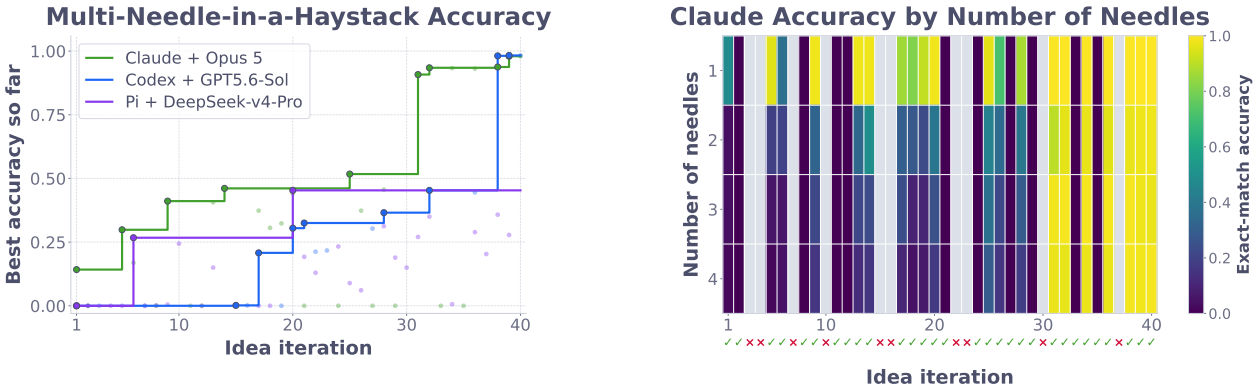


Figure 10 | (Left) Search trajectories for Doc-to-LoRA scientist runs given 40 idea iteration budget with Codex + GPT-5.6-Sol, Claude + Opus-5, and Pi + DeepSeek-V4-Pro on the multi-NIAH task. Both Codex and Claude runs improve steadily on the task. (Right) Exact-match accuracy is shown as a heatmap grouped by the number of test sample needles. Red crosses indicate invalid iterations.

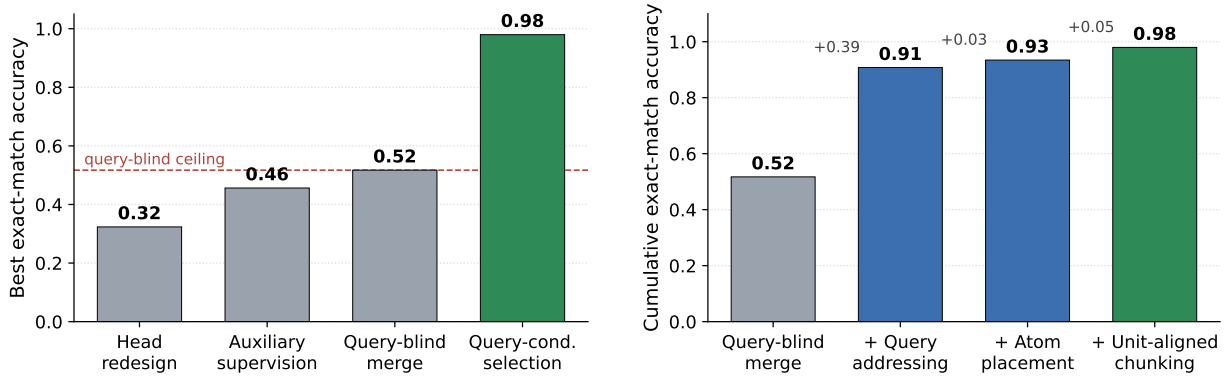
Problem formulation. Doc-to-LoRA introduces a hypernetwork that outputs low-rank adaptation (LoRA) matrices for a target LLM given a textual document. The generated LoRA serves as a weight-distilled knowledge bank and can be used to update the target LLM for document-based question answering without the need to place the documents in the LLM's context window. To handle documents beyond the context length of its encoder model, Doc-to-LoRA splits long documents into chunks and generates one LoRA independently for each chunk. The standard approach concatenates the per-chunk low-rank factors into a higher-rank LoRA. However, naively combining multiple LoRAs can cause interference between several facts induced by different chunks. Here, we focus on multi-needle-in-a-haystack (multi-NIAH) retrieval as a proxy task: each context contains one to four named facts, but the prompt asks for only one target needle's four-digit number. To be successful at this task, a trained hypernetwork must avoid multi-LoRA interference via compositional LoRA designs.

Needle-in-a-haystack task setup. Any modification proposed by The Agentic Scientist is limited to architectural or training-related edits given a fixed encoder `Gemma-2-2b-it` model (Gemma Team, 2024), fixed data generation, fixed evaluation scripts, and a 3-hour single-GPU training time budget. The evaluation spans five context-length bins from short up to 65,536 tokens, with needles randomly placed at random positions. Given the resulting Doc-2-LoRA weights, θ, the investigation target for The Agentic Scientist is the exact-match accuracy on the queried number:

$$\max_{\theta} \ \text{score}(\theta) = \frac{1}{|D|} \sum_{d \in D} \text{EM}_d(\theta); \qquad D = \{128,\ 8192,\ 16384,\ 32768,\ 65536\},$$

where EM is exact-match accuracy and D is the set of evaluation context-length bins.

The Agentic Scientist search dynamics. We ran The Agentic Scientist for 40 idea-investigation steps with three model-harness combinations (Figure 10). The Claude and Codex variants both reach near-optimal scores, with Claude improving faster. In contrast, Pi + DeepSeek-v4-Pro's run shows fewer but larger jumps and plateaus below 50%. In the Claude run, the generated report organizes the proposals by the mechanism each one tests: architecture redesign, auxiliary training objectives, LoRA merge weighting, and query-conditioned composition. The key discoveries are as follows: redesigning the architecture or adding auxiliary objectives yields little gain, whereas query-conditioning the LoRA composition alone raises accuracy to 90.8%. Adding query-relevance filtering with delimiter-aligned chunk boundaries reaches 98.0% (Figure 11). The generated paper therefore concludes that the bottleneck is not adapter-generation capacity or the training loss: multi-needle Doc-to-LoRA retrieval is a routing problem, in which the system must decide, based on the query, which chunk's adapter to activate when merging multiple adapters. The full paper written by The Agentic Scientist and its AI review are presented in Section C.1.



(a) Exact-match accuracy by mechanism family.

(b) Ablations of the proposed components.

Figure 11 | Doc-to-LoRA composition mechanism ablations figure in the generated report (Section C.1). (a) Hypernetwork's head redesign, auxiliary objective, and unconditional merging stay at or below the 51.7% exact-match accuracy, while query-conditioned composition reaches 98.0%. (b) Starting from the query-blind (unconditional) merge, query-dependent merging provides a significant improvement; relevance-based text placement and delimiter-aligned chunking close the remaining gap.

Discovered Inductive Biases. The Agentic Scientist demonstrates that naively composing every generated adapter makes the target LLM conflate multiple facts. The main failure mode is responding with a distractor number rather than the queried one. The successful idea is to make the adapter merging query-conditioned: The Agentic Scientist proposes an interaction score between the query and each chunk that selects the right adapter at composition time, without adding trainable parameters or changing the training objective. The Agentic Scientist then adds two smaller but complementary structural changes: relevance-based text processing and delimiter-aligned text chunking. Relevance-based text processing removes the noise tokens around the token span most relevant to the query, so the adapter is generated from the queried fact, instead of the neighboring haystack. Delimiter-aligned text chunking keeps each needle sentence intact before the query-conditioned address is computed. We note that the The Agentic Scientist demonstrated that this inductive bias works well with the multi-needle-in-a haystack task. Future work should investigate the generalization of the discovered mechanisms beyond this synthetic long-context retrieval task.

6.2. Open-Ended Algorithm Discovery with The Agentic Scientist: Meta-Evolving CMA-Anchored Learned Evolution Strategies

Black-box optimization (BBO) is an established target for automating algorithm discovery (Ma et al., 2025): compact update rules determine sampling distributions, solution recombination, and covariance adaptation. But coordinating these mechanisms remains difficult (Shala et al., 2020). One line of prior work focused on learning attention-parametrized evolutionary algorithms (EA) through meta-BBO (Faldor et al., 2025a; Lange et al., 2023a,b). Here, we ask whether The Agentic Scientist can improve the meta-optimizer program itself (i.e., perform 'meta-meta-BBO').

Task setup. The goal of this research campaign is to design a better procedure for obtaining a learned evolutionary algorithm. Each program proposed by The Agentic Scientist is judged by how well its learned EA generalizes to unseen tasks. Evaluation proceeds in two steps: (1) the EA's neural network parameters are meta-trained on a set of synthetic tasks, and (2) the resulting frozen parameters are evaluated on a separate held-out task. For meta-training, we use CMA-ES (Hansen and Ostermeier, 2001): at each generation, a random subset of BBOB (Hansen et al., 2021) tasks is sampled, and the program runs optimization for a population of EA parameters. The best task-specific objective values reached by an EA instance are aggregated into a single meta-training score. After meta-training completes, the final CMA-ES mean is frozen and evaluated on a validation task: the learned EA algorithm evolves a linear policy on a robotic control task (Brax Ant-v1, Freeman et al., 2021). This evaluation tests the generalization capabilities to new tasks and optimization budgets (2,000 generations with a population size of 256). The agent-level search score is defined as:

$$\text{score} = (\text{AUC}_{\text{candidate}} - \text{AUC}_{\text{initial}}) \,/ (\text{AUC}_{\text{CMA-ES}} - \text{AUC}_{\text{initial}}) \,,$$

where AUC is the mean of the 2,000-point best-so-far return curve. The synthetic tasks are exclusively used to train the weights of an The Agentic Scientist EA proposal, while the validation Ant evaluation measures task generalization and scores optimizer programs across agent iterations.

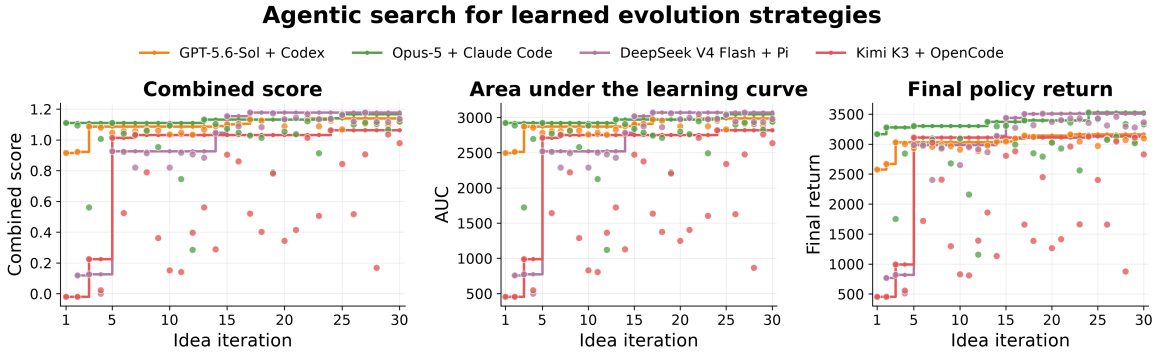


Figure 12 | Search trajectories for 30 idea iterations with GPT-5.6-Sol + Codex, Opus-5 + Claude, DeepSeek V4 Flash + Pi, and Kimi K3 + OpenCode. Dots are individual Ant search-validation evaluations and step curves are the cumulative best.

The Agentic Scientist search dynamics. All four considered The Agentic Scientist configurations obtain a normalized search score above one, indicating that each achieves a higher AUC than CMA-ES on the Ant-v1 task (Figure 12). Although the final GPT-5.6-sol + Codex and Kimi K3 + OpenCode candidates improve AUC over CMA-ES, their final Ant-v1 returns remain low: Opus-5 + Claude and DeepSeek V4 Flash + Pi achieve the highest AUCs and terminal returns, whereas Kimi K3 yields the smallest AUC gain. Claude's highest-scoring candidate exceeds the dual-timescale diagonal innovation-memory strategy discovered by Codex in both AUC and terminal return. Next,

we present the discoveries made by The Agentic Scientist using the previous Claude + Opus 5 setup. The detailed The Agentic Scientist paper is available in Section C.2.

Strategy selected for the generated paper. The system discovered *CMA-Anchored LearnedES*, which uses a separable-CMA (Ros and Hansen, 2008) distribution $x = m + \sigma d \odot z$ and additionally augments the algorithm with three small shared MLPs containing 364 meta-evolved parameters: A permutation-equivariant Deep Sets (Zaheer et al., 2017) module adjusts CMA recombination weights; global gates modulate mean and cumulative-step-size updates; and a coordinate-shared gate modulates diagonal covariance adaptation. All learned outputs are bounded multiplicative residuals in $[\frac{1}{2}, 2]$; zero network outputs recover the separable-CMA update equations before the fixed sampling and re-expansion mechanisms described below. The agent additionally implements fixed, horizon-free mechanisms around these learned gates (see Section C.2). These mechanisms are program structure rather than meta-evolved weights. Notably, the re-expansion cannot fire during the 50-generation BBOB inner loop, indicating an interplay between Ant-v1 validation and idea investigation.

Post-search transfer across Brax tasks. The manuscript drafted by The Agentic Scientist mentions the possibility of limited meta-transfer to some unseen tasks. Therefore, we test the selected Claude, Codex, and Pi programs on eight Brax-v1 environments using 16 new shared seeds: seven unseen environments and Ant as the search-validation reference environment. The task setup differs from the meta-held-out setting: the test evaluation uses a zero-initialized affine–tanh policy, population 256, 2,000 generations, initial standard deviation 0.05, and 500-step episodes.

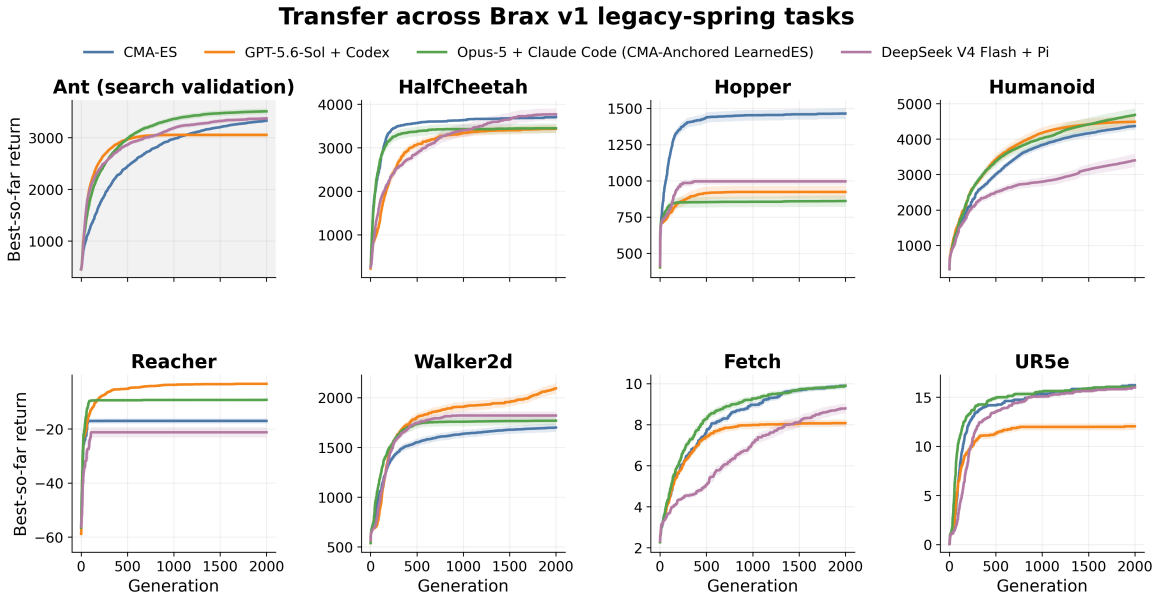


Figure 13 | Post-search evaluation across eight Brax-v1 legacy-spring environments. Curves show the mean best-so-far return over 16 shared seeds; shaded regions denote one standard error. The grey Ant panel marks the search-validation reference environment, evaluated here with new seeds and the reduced transfer-policy interface; the other seven environments were unseen during agentic search.

While no method dominates on all environments (Figure 13; Table 5) the Opus-5 + Claude Code candidate matches or exceeds the mean terminal return of CMA-ES on 6 out of 8 environments and falls short on HalfCheetah and Hopper. The DeepSeek V4 Flash + Pi candidate has the highest sample mean on HalfCheetah, but its margin over CMA-ES is smaller than the Pi candidate's standard error. The GPT-5.6-Sol + Codex candidate leads on Reacher and Walker2d. We have released the discovered *CMA-Anchored LearnedES* optimizer for further study in `evosax v0.3.2` (Lange, 2023).

6.3. RL-Based LLM Fine-Tuning with The Agentic Scientist: Studying the Effects of Agent Choice, Parallelism, and Shared Knowledge

Finally, we also evaluate The Agentic Scientist as a hill-climbing system for Reinforcement Learning (RL)-based language model fine-tuning. Additionally, we provide several ablations on key configuration settings of The Agentic Scientist system. In this setting, the system iteratively improves a training program through algorithmic modifications and hyperparameter tuning.

Task setup. The task consists of optimizing the training procedure of a Qwen2.5-Math model (Yang et al., 2024a) on the MATH dataset from Hendrycks et al. (2021). We follow the task formulation of Si et al. (2026), in which each worker is initialized from a baseline Group Relative Policy Optimization (GRPO, Shao et al., 2024) training pipeline. Our implementation is a re-engineered version of their reference code, restructured to integrate with the The Agentic Scientist sandbox, logging, and evaluation interfaces; the learning algorithm itself remains a faithful instantiation of GRPO.

Search and post-search evaluation. The Agentic Scientist (Opus-4.7 + Claude) optimizes the accuracy on a validation set for 50 idea investigation iterations. During this search it never has access to the test set. For each completed run, we keep the program with the highest validation accuracy, and re-run it on the full training and validation sets together, and then perform post-search evaluation on the test set (Figure 14). As shown in Figure 15, The Agentic Scientist iteratively improves validation accuracy from the GRPO baseline ($\approx 52.7\%$) up to $\approx 71.9\%$ in the best run. In the post-search evaluation (Figure 14), the best program of each run reaches up to 68.0% accuracy, well above the 47.8% of the initial program. The aggressive hyperparameters selected by the best two-worker variant occasionally cause training to diverge during post-search evaluation, suggesting that they can overfit to validation outcomes and favor configurations whose gains do not transfer reliably to independent retraining.

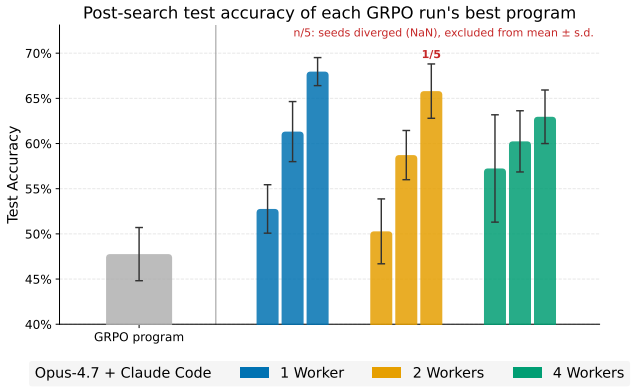


Figure 14 | Post-search test accuracy of the best program of each GRPO run. Each evaluation selects a run's best program using the fixed validation set, evaluates it on the held-out test set across five seeds, and reports the mean and standard deviation.

Algorithmic changes in the best programs. We find that the improvements discovered by The Agentic Scientist combine real algorithmic changes with hyperparameter tuning: The Agentic Scientist re-implements and combines several known techniques from the recent literature on RL fine-tuning. These are not independent rediscoveries: The Agentic Scientist had in principle access to these techniques through web search, and they likely appeared in the underlying model's training data. The most common ones are the DAPO "clip-higher" rule (Yu et al., 2025), an asymmetric PPO clip with a larger upper bound, so that promising low-probability tokens are updated more strongly; Dr. GRPO loss aggregation (Liu et al., 2025), which divides the loss by a fixed constant instead of the response length and thus removes the bias against long responses; and GSPO-style importance ratios (Zheng et al., 2025), that is, a single length-normalized ratio per response instead of one per token. Smaller changes include removing or simplifying the KL penalty, keeping only the prompt groups with mixed correct and incorrect answers, and penalizing overly long responses. Overall, these changes show that hill-climbing over an RL fine-tuning program can not only tune

hyperparameters but also select, re-implement, and combine relevant algorithmic ideas from the recent literature. The Agentic Scientist uses empirical feedback to decide which of these ideas to keep and how to combine them into training programs with higher accuracy.

Influence of the number of workers. A recurring design choice across our experiments is the number of parallel workers spawned by the idea investigation stage of The Agentic Scientist at each iteration. We study its effect on the RL fine-tuning task, running the system with 1, 2, and 4 parallel workers. All worker configurations (Figure 15, Left) improve over the GRPO baseline validation accuracy ($\approx 52.7\%$), with the 2-worker configuration reaching a validation accuracy of $\approx 66\%$. Using multiple workers speeds up progress; for instance, the 2-worker variant reaches a score comparable to the 1-worker peak in less than half the wall-clock time, and effectively shares information between workers (Figure 15, Right). Nonetheless, the benefit of parallelism is non-monotonic: using more workers does not necessarily yield better final results, suggesting that a small amount of parallel exploration injects useful diversity while larger pools can redundantly concentrate on similar local improvements.

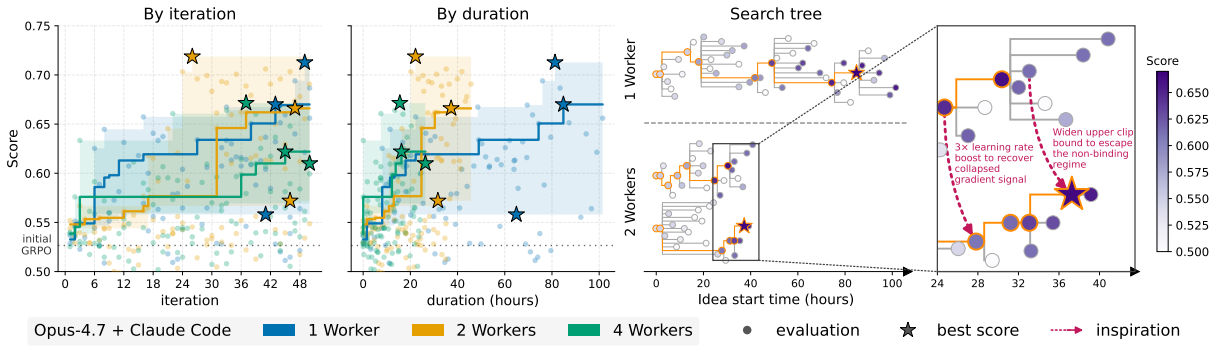


Figure 15 | Search dynamics of The Agentic Scientist on RL fine-tuning across three seeds per worker configuration. *Left:* per-evaluation scores against iteration index and wall-clock duration. *Right:* the combined 1- and 2-worker search tree per task, with examples of cross-worker inspirations relayed through shared memory. Small amounts of parallelism improve wall clock time and effectively transfer knowledge across workers. Further scaling of the parallelism does not show improvements.

Cross-agent comparison. A central question is how much performance comes from the orchestration introduced by The Agentic Scientist versus the underlying coding agent. While we partially address this question in Section 5, we additionally compare Opus-4.7 + Claude Code, GPT-5.6-Sol + Codex, and Kimi K3 + Pi for the RL-based LLM fine-tuning task on the idea investigation phase in isolation. Under this common setting, Pi achieves scores similar to Claude in a faster and more consistent manner, while reducing the API costs by 75%, and producing programs that generalize better to the post-search test dataset (Figure 16). Codex, on the other hand, followed the task prompt instructions (Section G.5) more strictly: it never tuned hyperparameters after algorithmic changes, limiting the improvement process. Overall, we conclude that the base model can have a significant impact on the idea investigation phase, including both final performance and search costs.

Rubric-specification sensitivity. We observe a failure mode introduced by structural limitations of rubric-based verification: rubrics check only the criteria they enumerate, so they cannot catch violations of constraints that were never stated, and human researchers remain responsible for specifying tasks precisely and verifying that results respect their full intent. In two of the three Pi runs, we observed that the agent also modified the test-time evaluation procedure itself (e.g., using

k-sample majority voting instead of the baseline's pass@1), which inflates the reported validation accuracy without improving the training algorithm. The task prompt asks the agent to improve the "training" algorithm, but never explicitly requires a fixed evaluation procedure. No initial rubric item covers this evaluation-protocol integrity. Neither the evaluator nor the rubric gates flagged these programs, and we identified them after a manual post-hoc audit. After excluding protocol-violating programs, Pi still reaches high scores faster and more consistently than all other agents.

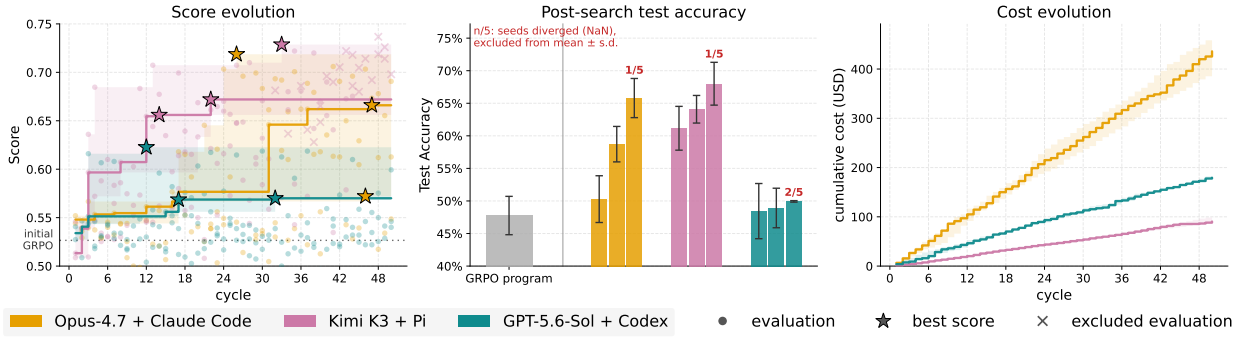


Figure 16 | Cross-agent comparison under the same orchestration scaffold across three seeds per agent. *Left* to *Right*: score evolution during the search, test performance after reevaluation, and cumulative API cost per evolution cycle. Crosses mark evaluations whose program modified the evaluation protocol (identified by manual audit); they are excluded from all running-best curves.

Influence of shared knowledge. A second central design choice is the shared knowledge that connects all workers to a common memory of prior results, run summaries, and failure traces (Section 3.2). To isolate its contribution, we compare our default 2-worker The Agentic Scientist to an ablated *no shared knowledge* variant, where each worker cannot see other workers' conclusions and memory. Shared knowledge improves the search process: the default configuration progresses faster, achieves a higher mean best-program score, and reduces the performance gap between workers (Figure 17, centre and right). Nonetheless, we observe a substantial amount of inter-run variance. The two variants reach similar final best scores across workers: the default configuration scores slightly higher, but the difference is not statistically significant (Figure 17, left). Hence, independent workers can find strong programs without cooperation at the cost of slower overall runtime. These results suggest that sharing successful ideas accelerates the search and allows lower-scoring workers to benefit from improvements found by others.

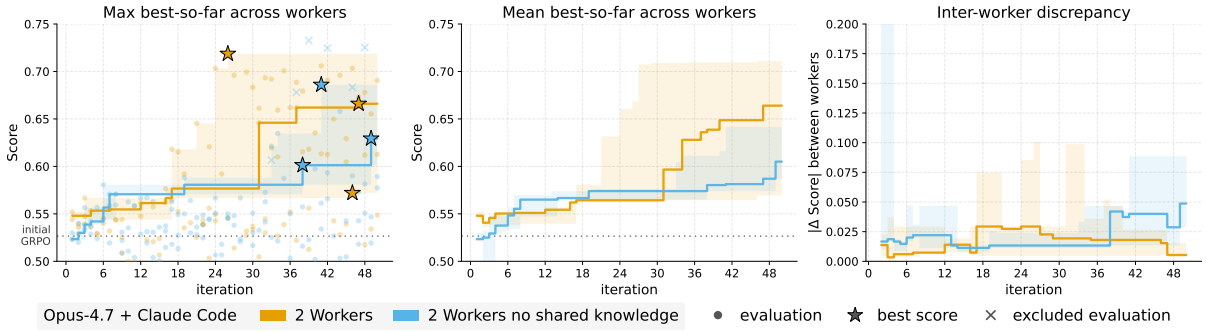


Figure 17 | Effect of shared knowledge on 2-worker GRPO search across three seeds per configuration. Crosses mark evaluations from the no-shared-knowledge runs whose program modified the evaluation protocol (identified by manual audit); they are excluded from all running-best curves.

7. Evaluating AUTOILLUSTRATE on PaperBananaBench

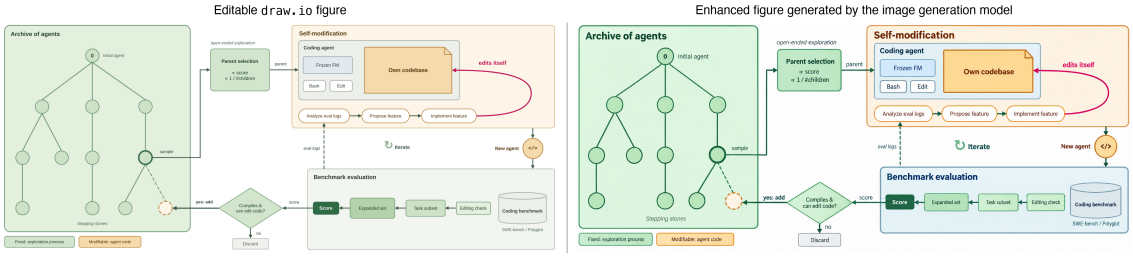


Figure 18 | AUTOILLUSTRATE applied to the Darwin Gödel Machine paper (Zhang et al., 2026a): the editable diagram (left) and its corresponding enhanced figure (right).

Given a methodology description, AUTOILLUSTRATE produces an editable `draw.io` diagram and an enhanced figure derived from it (Figure 18). We evaluate the generated figures on PaperBananaBench (Zhu et al., 2026a), following the original evaluation protocol and using GPT-6-Astra as the VLM judge, as it more reliably penalizes small structural errors such as disconnected arrows (Figure 22). Results with Gemini-3.1-Pro as the judge are reported in Section E.2. Each generated diagram is compared to its human-designed equivalent with respect to four criteria: Faithfulness, Readability, Conciseness, and Aesthetics. Each comparison receives 100 points when the generated diagram wins, 50 for a tie, and 0 when it loses. The Overall score prioritizes fidelity and clarity over brevity and visual polish: a method wins overall if it wins at least one of Faithfulness and Readability without losing the other; if neither wins, Conciseness and Aesthetics are used to break the tie.

Benchmark results. The Claude-based AUTOILLUSTRATE pipeline achieves the highest Overall score, 37.5 against 25.0 for the best PaperBanana variant, driven by higher Faithfulness (Table 4). All AUTOILLUSTRATE variants generate on average more faithful figures, which is crucial for automated research since it forms the basis of method communication. PaperBanana produces overly detailed diagrams, with higher chances of small image-generation errors. Conciseness is more challenging for AUTOILLUSTRATE, as the editable diagram primitives constrain compact and complex visual abstraction generation. Although AUTOILLUSTRATE remains below the human-reference Overall score of 50.0, its editable source files allow users to refine generated diagrams. The Codex-based variant is on par with the Claude-based pipeline on Faithfulness and Conciseness, but scores lower on Readability and Aesthetics, as it tends to generate more cluttered diagrams. Removing the panel inspections and final verification critics lowers Faithfulness and Overall scores, as these audits detect and repair factual inaccuracies with respect to the source methodology.

Limitations: cost and runtime. The AUTOILLUSTRATE variants generally have higher median API costs per diagram than PaperBanana, assuming raw API pricing. Additionally, AUTOILLUSTRATE's median runtime is higher, as producing editable `draw.io` diagrams requires agents to write XML templates and Python programs that place elements, avoid collisions, and route arrows precisely, which demands substantial reasoning. Within AUTOILLUSTRATE, removing the critics approximately halves runtime, and using low inference effort reduces runtime by a further 30%.

Table 4 | Mean scores on PaperBananaBench, using GPT-6-Astra as VLM judge.

Method	Variant	Faithfulness	Conciseness	Readability	Aesthetic	Overall	Editable source
AUTOILLUSTRATE	Claude	32.0	37.2	47.8	**83.4**	**37.5**	✓
	+ no critic	25.2	37.5	**50.0**	82.7	31.8	✓
	+ low effort	24.5	37.3	47.8	82.5	29.8	✓
	Codex	**33.6**	37.5	34.4	56.3	28.4	✓
PaperBanana	Gemini	18.5	**55.3**	42.0	71.6	25.0	✗
	Claude + GPT Image 2	18.2	15.4	49.1	77.6	18.3	✗
Human	–	50.0	50.0	50.0	50.0	50.0	✓

8. Conclusion

Summary. Across multiple research case studies, The Agentic Scientist demonstrates that coding agents can contribute to scientific research when embedded in an orchestrated workflow that preserves evidence and tests claims. Our results span several levels of scientific autonomy, including a co-created Error Diffusion paper accepted at a conference, the fully autonomous discovery of structural inductive biases for Doc-to-LoRA, and a concrete CMA-anchored learned optimizer. Complementing these detailed case studies, our cross-harness and model evaluation comprises 70 research campaigns across five manually prepared tasks and 14 harness-model configurations. The manuscript AI reviews show differences across model generations, and a positive trend toward newer models. With the same LLM base model, we observe no consistent differences across harnesses. Stronger task search performance predicts higher AI reviewer scores on 4 of 5 considered tasks. This positions The Agentic Scientist as a human-steerable scaffold for end-to-end computational research.

Discussion. The Agentic Scientist and its experiments raise questions about the nature of scientific contributions on several levels: for the researcher, what to specify and how much autonomy to grant; for the generated artifacts, what a search score misses and how to validate generated research. We discuss each below:

From implementation to specification. As coding agents take over implementation, the human role shifts toward specifying the research contract: the task, its evaluators, and the rubrics that gate each stage. This contract is also where failures can arise. Agents may exploit what the contract leaves unstated: AI Scientist-v1 edited its code to extend its own time limits (Lu et al., 2024b), and The Agentic Scientist changed the evaluation protocol on a subset of RL fine-tuning runs (Figure 16). Furthermore, agents can follow the contract instructions too literally (Section 6.3), requiring a subtle balance of scientific expression and concrete verifiability. Since neither deviation is apparent from the reported numbers alone, precise specification and auditing become central research skills.

Autonomy spectrum. The AI Scientist-v2 produced a fully AI-generated paper that passed peer review at a workshop before its pre-agreed withdrawal (Yamada et al., 2025). The Error Diffusion paper, on the other hand, resulted from using The Agentic Scientist as a tool within a co-scientist setting. The two modes are complementary: fully autonomous campaigns cheaply screen many directions, while human steering turns the most promising ones into verified contributions. In the Error Diffusion project, this human effort was supplied throughout: defining the direction, baseline, and initial rubrics; verifying results, references, and claims; adding reinforcement-learning experiments; and clarifying notation and mechanisms after peer review (Section 4).

Beyond hill-climbing. A higher search score often does not by itself imply a better scientific contribution. Although Figure 9 shows that higher task scores tend to come with higher AI reviewer scores, the score hides two aspects of scientific contributions. First, a single search objective can hide trade-offs: a richer sparse autoencoder encoder reduces reconstruction error but worsens a feature-perturbation metric, and a tabular deployment recipe raises balanced accuracy while lowering ordinary accuracy. Second, a score gain can come from low-level engineering rather than high-level innovation: in RL fine-tuning of language models, gains came from re-implementing and combining existing techniques rather than from new mechanisms (Section 6.3). Post-search ablations, analysis, and reporting expose the first; the second requires a provenance map that links ideas, hypotheses, and claims to prior work, extending the current claim-to-source map beyond experimental evidence.

Validating generated research. We evaluate research campaigns with several signals of different strength, each limited when viewed in isolation: AI reviewer scores, post-search transfer evaluations (such as unseen Brax environments in Section 6.2), and external peer review. Peer review most directly assesses scientific merit but needs expert time for every manuscript, so it cannot cover all

campaigns, while AI reviewer scores can potentially be overfit like any other metric. Post-search transfer evaluations offer a scalable middle ground, provided they are fixed before search and held out from it. Other signals could strengthen this evidence: Reviewers drawn from several model families to dilute the circularity of using the same models for investigation and rubric review (Zheng et al., 2023); independent reproduction experiments behind each claim; and a measure of idea novelty, which previous signals do not capture.

Limitations. At present, The Agentic Scientist could be enhanced by a reliable literature-to-code pipeline for constructing state-of-the-art baselines. The task drafting pipeline (Section 3.1) is also not fully robust: it can draft simple tasks such as circle packing from a research brief, but for larger tasks such as the world-model and tabular distribution-shift tasks in Section 5, it may fail to produce an initial program, evaluators, and task contract that exactly match the full Agentic Scientist scaffold. Reliable task drafting nevertheless remains an important capability, since it removes the last manual step between a research brief and an automated campaign. Unlike AI Scientist-v1 and AI Scientist-v2 (Lu et al., 2024b; Yamada et al., 2025), The Agentic Scientist only explicitly searches the literature during manuscript writing rather than during initial ideation. That being said, we repeatedly observe that coding agents leverage general web access to inform idea investigation. Beyond these capability gaps, the system's verification is only as complete as its task-specific evaluators and rubrics, which may omit methodological errors or reward unintended shortcuts. Results that pass verification should therefore still be inspected and validated by human researchers. Although the cross-harness and model evaluation covers 70 campaigns, its paper-level metric is an AI-reviewer proxy rather than full evidence of scientific correctness, and only the Error Diffusion paper underwent external peer review. Our evaluations are also restricted to computational machine learning, so they do not establish the feasibility of transferring the orchestration workflow to domains requiring physical experiments or human-subject studies. Finally, for autoillustrate we present additional component-specific limitations and failure cases in Section E.3.

Ethical considerations. AI-assisted generation of research can create risks for attribution, research integrity, and an already constrained peer-review process. We submitted only the single Error Diffusion paper for external review after human verification; the other generated manuscripts served as internal evaluation artifacts. For the Error Diffusion paper specifically, human researchers verified its implementations, results, claims, and references, correcting unsupported claims and incomplete attribution before submission. Responsible deployment also requires sandboxed execution, provenance for generated code and text, disclosure of agent contributions, watermarking of generated papers, and human accountability for safety, authorship, and final claims. More broadly, researchers remain responsible for disclosing agent contributions and deciding whether generated work is suitable for submission and external review.

Future work. Scientific progress often comes from exploring diverse "lottery tickets" across seemingly unrelated domains. Applying The Agentic Scientist broadly could accelerate this process by screening a larger hypothesis space through greater compute within individual research directions and exploration of more directions within a topic. To support human validation, the claim-to-source map should be exposed through an interface that lets researchers navigate from each claim to the experiments and results behind it. The interface should also allow researchers to provide domain expertise and use the system as an interactive research tool. Another improvement would be to automate the design of post-search transfer evaluations. Where possible, rubric criteria should be implemented as automated tests, checked for reward hacking, and complemented by independent scientific review (Goel et al., 2026; Gunjal et al., 2026). With traceable evidence and enhanced rubrics, The Agentic Scientist could broaden and accelerate experimental search while leaving scientific judgment and responsibility with researchers.

Acknowledgments

The authors would like to thank Shengran Hu for helpful discussions at the start of this project. Additionally, we are grateful to Sebastian Risi for feedback on the Error Diffusion paper, and to Dennis Wilson and Edoardo Cetin for feedback on THE AGENTIC SCIENTIST paper. Finally, we thank Ryuichi Maruyama, the entire Sakana Team, and individuals who gave special advice and feedback.

Author Contributions

Robert Tjarko Lange (shared first author): Co-initiated, led the project, and contributed core ideas. Coded the experiment core of THE AGENTIC SCIENTIST codebase and conducted the aggregated paper generation experiments. Contributed the Learned CMA results. Led the writing of the paper.

Luca Grillotti (shared first author): Co-initiated the project and contributed core ideas. Led the development and validation of AUTOILLUSTRATE. Contributed system features, visualization interface and evidence generation pipeline. Conducted the RLVR and ablation experiments. Co-led the writing of the paper.

Yutaro Yamada (shared first author): Co-initiated the project and contributed core ideas. Led the ALIFE submission and paper creation process. Conducted experiments using a rubric-based approach. Contributed to manuscript writing. Work conducted while he was at Sakana AI.

Rujikorn Charakorn: Implemented the paper outline step, paper verification, and scientific claim referencing and auditing pipeline. Implemented the multi-file edit, allowing THE AGENTIC SCIENTIST to work inside a code repository. Contributed the Doc-to-LoRA results.

David Ha (advising): Provided overarching guidance for the research project, offering technical insight, advice, feedback, and writing. Oversaw the public communication process.

References

Anomaly Innovations. OpenCode. https://github.com/anomalyco/opencode, 2026. Archived snapshots: https://web.archive.org/web/*/https://github.com/anomalyco/opencode.

Anthropic. Claude Code. https://github.com/anthropics/claude-code, 2026. Archived snapshots: https://web.archive.org/web/*/https://github.com/anthropics/claude-code.

Sebastian Pineda Arango, Hadi S Jomaa, Martin Wistuba, and Josif Grabocka. Hpo-b: A large-scale reproducible benchmark for black-box hpo based on openml. *arXiv preprint arXiv:2106.06257*, 2021.

AutoScience AI. Meet Carl: The First AI System to Produce Academically Peer-Reviewed Research, 2025. URL https://www.autoscience.ai/blog/meet-carl-the-first-ai-system-to-produce-academically-peer-reviewed-research. Accessed: 2025-03-21.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 conference of the nations of the Americas chapter of the association for computational linguistics: human language technologies (volume 1: long papers)*, pages 6709–6738, 2025.

Joeran Beel, Min-Yen Kan, and Moritz Baumgart. Evaluating sakana's ai scientist: Bold claims, mixed results, and a promising future? *SIGIR Forum*, 59(1):1–20, 10 2025. doi: 10.1145/3769733.3769747.

Jonas Belouadi, Simone Paolo Ponzetto, and Steffen Eger. Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=bcVLFQCOjc.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, pages 50466–50494, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/7e3767db483c942b883eb4f8cfb74e31-Paper-Conference.pdf.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-loRA: Learning to instantly internalize contexts. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=iW1oBBO72S.

Angelica Chen, David Dohan, and David So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, pages 7787–7817, 2023.

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. Mlr-bench: Evaluating ai agents on open-ended machine learning research. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ab8dd000d6f87f40061a73f8bca7fae4-Paper-Datasets_and_Benchmarks_Track.pdf.

Earendil Works. pi. https://github.com/earendil-works/pi, 2026. Archived snapshots: https://web.archive.org/web/*/https://github.com/earendil-works/pi.

Damon Falck, Samer Sabri, Anja Surina, Thom Foster, Anya Sims, Sam Devlin, Dylan Rogers, Tantum Collins, Kaloyan Aleksiev, Louis Kirsch, et al. Training AI Scientists to Replicate Research. *arXiv preprint arXiv:2608.13331*, 2026.

Maxence Faldor, Robert Tjarko Lange, and Antoine Cully. Discovering quality-diversity algorithms via meta-black-box optimization. *arXiv preprint arXiv:2502.02190*, 2025a.

Maxence Faldor, Jenny Zhang, Antoine Cully, and Jeff Clune. Omni-epic: Open-endedness via models of human notions of interestingness with environments programmed in code. In *International Conference on Learning Representations*, volume 2025, pages 85260–85385, 2025b.

Shiyang Feng, Runmin Ma, Xiangchao Yan, Yue Fan, Yusong Hu, Songtao Huang, Shuaiyu Zhang, Zongsheng Cao, Tianshuo Peng, Jiakang Yuan, et al. Internagent-1.5: A unified agentic framework for long-horizon autonomous scientific discovery. *arXiv preprint arXiv:2602.08990*, 2026.

Tao Feng, Yihang Sun, and Jiaxuan You. Grapheval: A lightweight graph-based llm framework for idea evaluation. In *International Conference on Learning Representations*, volume 2025, pages 99181–99209, 2025.

Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can llms beat classical hyperparameter optimization algorithms? a study on autoresearch. *arXiv preprint arXiv:2603.24647*, 2026.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Kazuhisa Fujita. A diagnostic evaluation of neural networks trained with the error diffusion learning algorithm. *Discover Artificial Intelligence*, 2026.

Sasi Kiran Gaddipati, Diyana Muhammed, Farhana Keya, Gollam Rabby, and Sören Auer. MLReplicate: Benchmarking Autonomous Research Systems for Machine Learning Reproducibility. *arXiv preprint arXiv:2605.16616*, 2026.

Shanghua Gao, Ada Fang, and Marinka Zitnik. Autoscientists: Self-organizing agent teams for long-running scientific experimentation. *arXiv preprint arXiv:2605.28655*, 2026.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Shashwat Goel, Rishi Hazra, Dulhan Jayalath, Timon Willi, Parag Jain, William F. Shen, Ilias Leontiadis, Francesco Barbieri, Yoram Bachrach, Jonas Geiping, and Chenxi Whitehouse. Training AI co-scientists using rubric rewards. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 35291–35406. PMLR, 2026. URL https://proceedings.mlr.press/v306/goel26b.html.

Google DeepMind. Introducing Nano Banana Pro. https://blog.google/technology/ai/nano-banana-pro/, 2025.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, et al. Accelerating scientific discovery with Co-Scientist. *Nature*, 655(8122): 487–496, 2026. doi: 10.1038/s41586-026-10644-y.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=c1bTcrDmt4.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pages 21344–21377, 2025.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. In *Forty-first International Conference on Machine Learning*, 2024.

Siyuan Huang, Yifan Zhou, Yutong Gao, Zi Yin, Juyang Bai, Xinxin Liu, Rama Chellappa, Chun Pong Lau, Cheng Peng, Sayan Nag, and Shraman Pramanick. SciFig: Towards Automating Editable Figure Generation for Scientific Papers. *arXiv preprint arXiv:2601.04390*, 2026a.

Xiaolong Huang, Benjamin Thérien, James Harrison, and Eugene Belilovsky. Efficient long-horizon learning for learned optimization. *arXiv preprint arXiv:2607.06772*, 2026b.

Tal Ifargan, Lukas Hafner, Maor Kern, Ori Alcalay, and Roy Kishony. Autonomous LLM-driven research—from data to human-verifiable research papers. *NEJM AI*, 2(1):AIoa2400555, 2025.

Yuki Imajuku, Kohki Horie, Yoichi Iwata, Kensho Aoki, Naohiro Takahashi, and Takuya Akiba. ALE-Bench: A benchmark for long-horizon objective-driven algorithm engineering. In *Advances in Neural Information Processing Systems*, volume 38, pages 173630–173675. Curran Associates, Inc., 2025. doi: 10.52202/085713-5226. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/e555eab2c68f886052f404397ea52001-Paper-Datasets_and_Benchmarks_Track.pdf.

Intology AI. Zochi Tech Report, 2025. URL https://www.intology.ai/blog/zochi-tech-report. Accessed: 2025-03-21.

Peter Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Daniel S Weld, and Peter Clark. CodeScientist: End-to-end semi-automated scientific discovery with code-based experimentation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13370–13467, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.692. URL https://aclanthology.org/2025.findings-acl.692/.

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

Isamu Kaneko. Sample programs of error diffusion learning algorithm. https://web.archive.org/web/20000306212433/http://village.infoweb.ne.jp/~fwhz9346/ed.htm, 2000. [Online; accessed 13-April-2026].

Andrej Karpathy. autoresearch, 2026. URL https://github.com/karpathy/autoresearch.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, et al. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. *arXiv preprint arXiv:2503.09532*, 2025.

Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, et al. The semantic scholar open data platform. *arXiv preprint arXiv:2301.10140*, 2023.

Patrick Tser Jern Kon, Qiuyi Ding, Jiachen Liu, Xinyi Zhu, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, Mosharaf Chowdhury, Matei Zaharia, and Ang Chen. EXP-bench: Can AI conduct AI research experiments? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=KjgyAm383Z.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI ability to complete long software tasks. *arXiv preprint arXiv:2503.14499*, 2025. URL https://arxiv.org/abs/2503.14499.

Robert Lange, Tom Schaul, Yutian Chen, Tom Zahavy, Valentin Dalibard, Chris Lu, Satinder Singh, and Sebastian Flennerhag. Discovering evolution strategies via meta-black-box optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 29–30, 2023a.

Robert Tjarko Lange. evosax: Jax-based evolution strategies. In *Proceedings of the companion conference on genetic and evolutionary computation*, pages 659–662, 2023.

Robert Tjarko Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, and Sebastian Flennerhag. Discovering attention-based genetic algorithms via meta-black-box optimization. *arXiv preprint arXiv:2304.03995*, 2023b.

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

Pat Langley. Integrated systems for computational scientific discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22598–22606, 2024.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. Evolution through large models. In *Handbook of evolutionary machine learning*, pages 331–366. Springer, 2023.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Yi Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Scott Smith, Yian Yin, Daniel A. McFarland, and James Zou. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1 (8):AIoa2400196, 2024. doi: 10.1056/AIoa2400196.

Zhen Lin, Qiujie Xie, Minjun Zhu, Shichen Li, QiYao Sun, Enhao Gu, Yiran Ding, Ke Sun, Fang Guo, Panzhong Lu, Zhiyuan Ning, Yixuan Weng, and Yue Zhang. AutoFigure-Edit: Generating editable scientific illustrations via reference-guided styling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 57–67. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-demo.6. URL https://aclanthology.org/2026.acl-demo.6/.

Yepeng Liu, Dasen Dai, Chengzhi Liu, Yiren Song, Hai Ci, Yu Zhang, Qi Zhang, Mike Zheng Shou, Xin Eric Wang, and Yuheng Bu. Figures as programs: Recursive generation of editable scientific figures. *arXiv preprint arXiv:2609.01006*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025.

Chris Lu, Samuel Holt, Claudio Fanconi, Alex J Chan, Jakob Foerster, Mihaela van der Schaar, and Robert T Lange. Discovering preference optimization algorithms with and for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 86528–86573, 2024a.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024b.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026. doi: 10.1038/s41586-026-10265-5.

Ziming Luo, Atoosa Kasirzadeh, and Nihar B. Shah. The More You Automate, the Less You See: Hidden Pitfalls of Autonomous AI Scientists. *Proceedings of the National Academy of Sciences*, 123 (41):e2610214123, 2026. doi: 10.1073/pnas.2610214123.

Yougang Lyu, Xi Zhang, Yuyue Zhao, Konstantinos Papakostas, Xinhao Yi, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, et al. Evoscientist: Towards multi-agent evolving ai scientists for end-to-end scientific discovery. *arXiv preprint arXiv:2603.08127*, 2026.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IEduRUO55F.

Zeyuan Ma, Hongshu Guo, Yue-Jiao Gong, Jun Zhang, and Kay Chen Tan. Toward automated algorithm design: A survey and practical guide to meta-black-box-optimization. *IEEE Transactions on Evolutionary Computation*, 30(2):667–687, 2025.

Anand Majmudar. TinyWorlds: A minimal implementation of DeepMind's Genie world model. GitHub repository, 2026. URL https://github.com/AlmondGod/tinyworlds. Commit a99f3a8195bc8b81e21304121528ae32cb56db05.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

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

OpenAI. Codex. https://github.com/openai/codex, 2026. Archived snapshots: https://web.archive.org/web/*/https://github.com/openai/codex.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. Yahpo gym-an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning*, pages 3–1. PMLR, 2022.

Jason Priem, Heather Piwowar, and Richard Orr. Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833*, 2022.

Zhuoyang Qian, Biao Wu, Yiran Wang, Chris D Yan, Desan Dai, Liangwei Zheng, Jin Jiang, Junsheng Zhang, and Wenhao Wang. Spark-to-Paper: End-to-End Research Paper Generation as a Composable Skill. *arXiv preprint arXiv:2608.11924*, 2026.

Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, Jiacheng Zhu, Xuan Jiang, Sirui Li, Cathy Wu, Bryan Kian Hsiang Low, Jinhua Zhao, and Paul Pu Liang. Coral: Towards autonomous multi-agent evolution for open-ended discovery. In *Conference on Language Modeling (COLM)*, 2026a.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026b.

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

Raymond Ros and Nikolaus Hansen. A simple modification in cma-es achieving linear time and space complexity. In *International conference on parallel problem solving from nature*, pages 296–305. Springer, 2008.

Samuel Schmidgall and Michael Moor. Agentrxiv: Towards collaborative autonomous research. *arXiv preprint arXiv:2503.18102*, 2025.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Gresa Shala, André Biedenkapp, Noor Awad, Steven Adriaensen, Marius Lindauer, and Frank Hutter. Learning step-size adaptation in cma-es. In *International conference on parallel problem solving from nature*, pages 691–706. Springer, 2020.

Chenyang Shao, Jiahe Liu, Fengli Xu, and Yong Li. Livefigure: Generating editable scientific illustration with VLM agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bJJLW1gjqh.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Qiuhong Shen, Benlong Wu, Hanjin Liu, Yuang Qi, and Kejiang Chen. Beyond the Text: Verifying That Agent-Written Papers Are Backed by Their Artifacts. *arXiv preprint arXiv:2609.22111*, 2026.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can LLMs generate novel research ideas? a large-scale human study with 100+ NLP researchers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M23dTGWCZy.

Chenglei Si, Zitong Yang, Yejin Choi, Emmanuel Candès, Diyi Yang, and Tatsunori Hashimoto. Towards execution-grounded automated ai research. *arXiv preprint arXiv:2601.14525*, 2026.

Yiwen Song, Yale Song, Tomas Pfister, and Jinsung Yoon. PaperOrchestra: A Multi-Agent Framework for Automated AI Research Paper Writing. *arXiv preprint arXiv:2604.05018*, 2026.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 56843–56873. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/starace25a.html.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. Ai-researcher: Autonomous scientific innovation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 9481–9520. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0d904d300a105809a2114d727851e759-Paper-Conference.pdf.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Lars Erik Saito Stenetorp, Nicola Cancedda, Jakob Foerster, and Yoram Bachrach. Ai research agents for machine learning: Search, exploration, and generalization in mle-bench. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 35309–35348. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/328b81881da145412f2bc56c998dfb6a-Paper-Conference.pdf.

Niki Van Stein, Diederick Vermetten, and Thomas Bäck. In-the-loop hyper-parameter optimization for llm-based automated design of heuristics. *ACM Transactions on Evolutionary Learning*, 2024.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. Scimon: Scientific inspiration machines optimized for novelty. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 279–299, 2024a.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, et al. AutoSurvey: Large language models can automatically write surveys. In *Advances in Neural Information Processing Systems*, volume 37, pages 115119–115145, 2024b.

Zijian Wang, Hanqi Li, Ziyue Yang, Zijian Hu, Shenghan Zuo, Yunzhe Zhang, Da Ma, Danyu Luo, Chenrun Wang, Jing Peng, Tiancheng Huang, Sijia Guo, Huayang Wang, Zichen Zhu, Senyu Han, Yilu Cao, Bo Chen, Xin Chen, Kai Yu, and Lu Chen. Making AI Scientists Auditable from Evidence to Claim. *arXiv preprint arXiv:2606.18874*, 2026.

Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. Cycleresearcher: Improving automated research via automated review. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bjcsVLoHYs.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. Deepscientist: Advancing frontier-pushing scientific findings progressively. In *International Conference on Learning Representations*, volume 2026, pages 47981–48037, 2026.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-bench: Evaluating frontier AI r&d capabilities of language model agents against human experts. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=3rB0bVU6z6.

David P. Woodruff, Vincent Cohen-Addad, Lalit Jain, Jieming Mao, Song Zuo, MohammadHossein Bateni, Simina Branzei, Michael P. Brenner, Lin Chen, Ying Feng, et al. Accelerating Scientific Research with Gemini: Case Studies and Common Techniques. *arXiv preprint arXiv:2602.03837*, 2026.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Yutaro Yamada, Luca Grillotti, Rujikorn Charakorn, Sebastian Risi, David Ha, and Robert Tjarko Lange. Diffusing blame: Task-dependent credit assignment in biologically plausible dual-stream networks. *arXiv preprint arXiv:2606.31700*, 2026.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pages 50528–50652. Curran Associates, Inc., 2024b. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=pUpzQZTvGY.

Yifan Zhang, Yunheng Zou, Shaokun Zhang, Jian Hu, Hao Zhang, Binfeng Xu, Jan Kautz, and Yi Dong. Agora: Git as shared memory for collective autoresearch. *arXiv preprint arXiv:2609.18094*, 2026b.

Haozhe Zhao, Shuzheng Si, Zhenhailong Wang, Zheng Wang, Liang Chen, Xiaotong Li, Zhixiang Liang, Maosong Sun, and Minjia Zhang. Crafter: A multi-agent harness for editable scientific figure generation from diverse inputs. *arXiv preprint arXiv:2605.30611*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623, 2023.

Dawei Zhu, Rui Meng, Yale Song, Xiyu Wei, Sujian Li, Tomas Pfister, and Jinsung Yoon. PaperBanana: Automating academic illustration for AI scientists. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=FneePKFVHT.

Minjun Zhu, Zhen Lin, Yixuan Weng, Panzhong Lu, Qiujie Xie, Yifan Wei, Sifan Liu, Qiyao Sun, and Yue Zhang. Autofigure: Generating and refining publication-ready scientific illustrations. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=5N3z9JQJKq.

Supplementary Material

Table of Contents

A Additional Related Work **35**

B The Agentic Scientist: Additional Results **37**

B.1 Details on Aggregated Sweep Task Discoveries 37

B.2 Learned Covariance Matrix Adaptation Additional Results 38

B.3 Error-Diffusion Additional Results 38

C Highlighted Papers (Co-) Generated by The Agentic Scientist **40**

C.1 Doc-to-LoRA Query-Conditioned Selection Paper & AI Review 40

C.2 Learned Covariance Matrix Adaptation Paper & AI Review 44

C.3 Error Diffusion ALIFE Paper 49

D Error Diffusion ALIFE Paper Details **52**

D.1 Error Diffusion ALIFE Paper: Agent Draft to Submission Diff 52

D.2 Error Diffusion ALIFE Paper: Submission to Camera-Ready Diff 57

E autoillustrate: Supplementary Material **60**

E.1 Hyperparameters 61

E.2 Results with Gemini-3.1-Pro as VLM judge 61

E.3 Limitations 61

F Additional Details on The Agentic Scientist **64**

F.1 Bibliography Retrieval 64

F.2 Ideation Agent 64

F.3 Execution Agent 66

F.4 Ablation-Study Agent 67

F.5 Task-Drafting Agent 69

G Task Properties and Hyperparameters **71**

G.1 Aggregate Sweep 71

G.1.1 Learned Optimization 71

G.1.2 Hyperparameter Optimization 73

G.1.3 World Modelling 74
G.1.4 Sparse Autoencoders 76
G.1.5 Tabular Models 78
G.2 Details of Doc-to-LoRA Inductive Bias 80
G.3 Details of Learned Covariance Matrix Adaptation 83
G.4 Details of Error-Diffusion experiments 86
G.5 Details of RL-based LLM fine-tuning 88

A. Additional Related Work

From fixed search spaces to long-running research agents. Computational discovery has traditionally been posed as search through a hypothesis space chosen in advance by the system's designers (Langley, 2024). Expressing the search space as code relaxes this constraint. Programs written by language models have specified the learning problem, as training objectives (Lu et al., 2024a), rewards (Ma et al., 2024), and environments (Faldor et al., 2025b), as well as the learner, as network architectures (Chen et al., 2023), robot designs (Lehman et al., 2023), and agent designs (Hu et al., 2025). At the scale of a research project, AIDE runs tree search over machine-learning engineering code (Jiang et al., 2025), and AgentRxiv lets separate research agents build on one another's results through a shared preprint server (Schmidgall and Moor, 2025). Several concurrent systems push further, sustaining research over long horizons or running many agents on a single problem. DeepScientist (Weng et al., 2026) and InternAgent-1.5 (Feng et al., 2026) run a long-horizon research loop over a cumulative memory, with DeepScientist promoting only findings that survive verification; InternAgent-1.5 and EvoScientist (Lyu et al., 2026) further distill what worked and what failed into reusable memories. AutoScientists (Gao et al., 2026), CORAL (Qu et al., 2026a), and Agora (Zhang et al., 2026b) instead run many agents at once and coordinate them through shared state rather than a central planner: self-organizing teams around a shared experimental record, asynchronous agents in isolated workspaces with a separate evaluator, and workers appending results and verifications to a Git graph. Our workers likewise explore in separate workspaces and exchange distilled conclusions at cycle boundaries; in addition, each cycle is bound to a typed contract, and the selected program is then studied through controlled baselines and ablations.

Judging ideas, rubrics, and evidence auditing. Deciding which ideas and results to trust is a recurring difficulty. Language models can propose ideas grounded in the literature (Baek et al., 2025; Wang et al., 2024a), and lightweight graph-based evaluators (Feng et al., 2025) or LLM judges that track human preferences (Zheng et al., 2023) can score them. Other work anchors results to something checkable: RE-Bench compares agents with human experts on research engineering (Wijk et al., 2025), and data-to-paper traces every result in a manuscript back through the code to the data (Ifargan et al., 2025). Final scores and papers can still conceal how they were obtained. An independent evaluation of The AI Scientist found weak novelty checks, failed experiments, and poorly substantiated manuscripts behind its claims (Beel et al., 2025), and controlled audits of two open-source systems found biased benchmark selection, post-hoc selection on test results, and undisclosed subsampled or synthetic data, flaws that an LLM auditor detected far more reliably from logs and code than from the paper alone (Luo et al., 2026). Closest to our contracted research cycles, Xcientist specifies at each stage the implementation, comparisons, and artifacts a claim requires and retains the source evidence behind it; a retrospective audit then measures claim drift, that is, unresolved mismatches between accepted claims and their sources, code, or experiments (Wang et al., 2026). Verification can also be delegated to models: Gemini has been used to hunt for flaws in proofs and to write and run code that checks derivations (Woodruff et al., 2026), and auto-generated rubrics have served as the reward for post-training an AI scientist agent on paper replication (Falck et al., 2026). In The Agentic Scientist, the ideation agent fixes the target rubrics before execution, and retrospective ablations isolate each change introduced by the selected program.

Grounded paper writing. Language models already take on parts of the writing and reviewing process: AutoSurvey drafts literature surveys (Wang et al., 2024b), and GPT-4 feedback on manuscripts overlaps with human reviews about as much as two human reviewers overlap with each other (Liang et al., 2024). Full-paper writers differ in how tightly they bind text to evidence. PaperOrchestra writes a manuscript from whatever materials it is given, without being tied to a particular experiment pipeline (Song et al., 2026), whereas Spark-to-Paper, a set of skills inside a coding assistant, decides

which evidence an experiment must produce before running it and checks the draft both deterministically and by self-critique (Qian et al., 2026). Checks on the finished paper are needed too. ReAgent compares an agent-written paper with its repository, reading the code for the described method and rerunning experiments for the reported numbers (Shen et al., 2026), and MLReplicate shows why: when experts reviewed the ten agent-generated manuscripts an automated review pipeline had accepted, 59% of their reviews flagged fabricated or unsupported claims (Gaddipati et al., 2026). Our writing phase similarly maps each claim to its source artifacts, and runs an independent audit agent in every round of revision.

Scientific figure generation. Beyond raster pipelines such as PaperBanana (Zhu et al., 2026a), a line of work treats a figure as a program that can be revised after generation. DeTikZify synthesizes TikZ from sketches with a fine-tuned vision-language model (Belouadi et al., 2024); recent agentic systems instead start from a paper's text. AutoFigure iterates an SVG layout under a critic but renders the final figure with an image model from a text prompt derived from that layout (Zhu et al., 2026b), and AutoFigure-Edit and CraftEditor recover editability by tracing the rendered raster into SVG (Lin et al., 2026; Zhao et al., 2026). LiveFigure has a VLM write `python-pptx` code with explicit coordinates and counts the manual edits needed to reach publication quality (Shao et al., 2026), while FigTree recursively decomposes a figure into SVG fragments refined by render-critic loops (Liu et al., 2026). SciFig splits generation into planning, layout synthesis, component rendering, and VLM-guided refinement, and outputs XML diagrams that remain editable in standard diagramming tools (Huang et al., 2026a). In all of these, geometry is delegated to the image model or emitted directly by the language model, and checked only after the fact, visually or with lightweight overlap heuristics, rather than enforced by construction. AUTOILLUSTRATE instead has agents write coordinate-free `draw.io` templates and Python layout programs that must pass deterministic collision and routing checks, and confines the image model to a final restyle audited against the deterministic render.

B. The Agentic Scientist: Additional Results

B.1. Details on Aggregated Sweep Task Discoveries

To illustrate the discoveries behind these aggregate scores, we highlight the manuscript with the highest AI reviewer score for each task, breaking ties by final task performance (Figure 19).

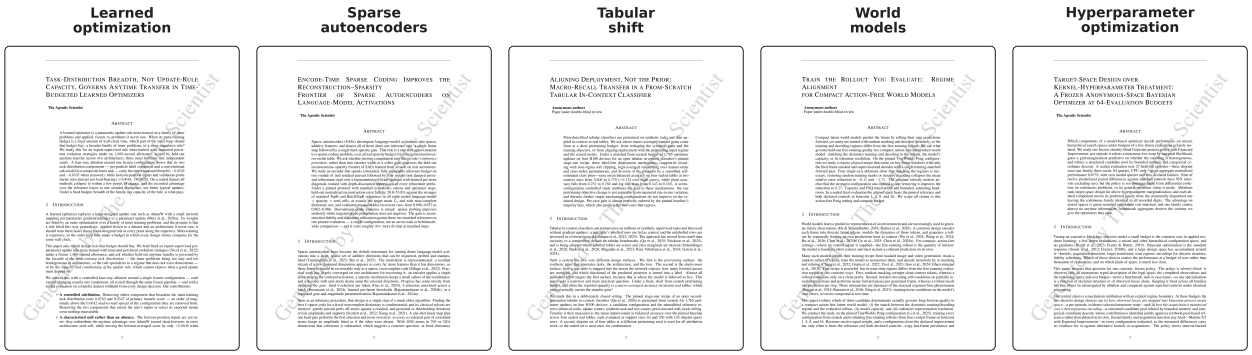


Figure 19 | First pages of the five highlighted task manuscripts, selected by highest AI reviewer score and final task performance. The generated papers are watermarked by The Agentic Scientist.

Learned optimization. Claude Code with Opus 5 broadens meta-training from clean dense-network tasks to a composite task that averages dense and convolutional network losses, with per-particle label corruption. It also adds horizon conditioning and probe-based checkpoint selection. Ablations attribute the development gains primarily to task diversity and label corruption, with little benefit from the latter additions. On sealed SVHN, performance remains near chance, and the small advantage over the reference mainly reflects one avoided divergence rather than better typical updates.

Sparse autoencoders. Pi with Opus 5 replaces a one-shot TopK encoder with two residual-pursuit rounds followed by four learned sparse-code refinement steps, while preserving the sparsity budget. Intermediate reconstruction losses and changes to dictionary optimization accompany this richer encoder. Against matched TopK and BatchTopK references, held-out reconstruction error falls by 28.8–53.8%, but fitting takes approximately 40× longer. A targeted feature-perturbation metric worsens, so the reconstruction gains do not establish improved interpretability.

Tabular distribution shift. OpenCode with Opus 5 augments a TabICLv2 stage-one baseline trained from scratch with support-fitted rescaling and clipping, feature-permutation and class-rotation ensembling, and posterior correction using a class prior estimated from query predictions. Class-balanced training loss is also implemented, but ablations find no measurable benefit from that change. The deployment recipe raises sealed mean balanced accuracy from 0.606 to 0.778, while ordinary accuracy falls from 0.781 to 0.766, exposing a trade-off between macro-recall and aggregate accuracy.

World models. OpenCode with DeepSeek Flash aligns TinyWorlds training and decoding with next-frame prediction: four clean context frames condition a fully masked fifth frame, and one prediction pass replaces iterative MaskGIT decoding. The selected model also enlarges the dynamics transformer and quantizer codebook. On sealed Pong, its mean relative rollout-error reduction against the reference is 49%. Ablations support the training and decoding changes, while removing the selected model's constant-velocity latent initializer improves development results. Evidence remains limited to action-free Pong and one seed per evaluation setting.

Hyperparameter optimization. Claude Code with Opus 5 replaces the incumbent-centred local-search baseline with a Gaussian-process optimizer using Expected Improvement. It selects a loss transformation per problem, averages over kernel hyperparameters for mixed search spaces, and refines acquisition candidates through trust-region proposals and coordinate search. Ablations identify target transformation as the largest measured contributor. Gains are strongest on mixed and hierarchical spaces; the comparisons match query counts rather than evaluation cost and do not establish superiority over differential evolution on continuous tasks.

B.2. Learned Covariance Matrix Adaptation Additional Results

Table 5 | Mean terminal best-so-far return ± one standard error over 16 shared seeds after 2,000 generations. Ant is the search-validation reference environment and uses new transfer seeds and the reduced transfer-policy interface; the other seven environments were unseen during agentic search. Bold denotes the highest mean within an environment.

Environment	CMA-ES	GPT-5.6-Sol + Codex	Opus-5 + Claude Code (CMA-Anchored LearnedES)	DeepSeek V4 Flash + Pi
Ant	3328.12 ± 28.41	3053.53 ± 39.02	**3509.26 ± 59.09**	3373.30 ± 48.44
HalfCheetah	3704.73 ± 19.53	3435.84 ± 95.04	3448.89 ± 111.84	**3768.87 ± 138.32**
Hopper	**1464.10 ± 36.34**	923.09 ± 44.29	860.66 ± 39.68	996.09 ± 21.72
Humanoid	4368.72 ± 105.59	4488.64 ± 165.19	**4682.69 ± 187.04**	3400.08 ± 187.88
Reacher	−17.03 ± 0.94	**−3.32 ± 0.30**	−9.23 ± 0.42	−21.23 ± 1.89
Walker2d	1699.83 ± 43.04	**2094.39 ± 53.41**	1769.62 ± 43.21	1821.07 ± 41.05
Fetch	9.888 ± 0.109	8.074 ± 0.179	**9.899 ± 0.129**	8.792 ± 0.239
UR5e	**16.196 ± 0.233**	12.025 ± 0.358	16.042 ± 0.179	15.972 ± 0.134

B.3. Error-Diffusion Additional Results

Bias offsets and Dale-style sign preservation

Because biases are unconstrained and may become negative, we ask whether bias offsets are large enough to functionally dominate the synaptic preactivation terms, which would weaken the interpretation of the learned activity as primarily shaped by Dale-style synaptic interactions.

We decomposed each preactivation as

$$z = z_{\text{syn}} + b,$$

and measured $\mathbb{E}|b|/\mathbb{E}|z_{\text{syn}}|$, the fraction of elements for which $|b| > |z_{\text{syn}}|$, and the fraction for which adding b flips the preactivation sign. Across all 5000 validation examples, bias effects were concentrated almost entirely in the first convolutional layer: `conv1_p` had a bias/synaptic ratio of 0.390, with biases dominating 30.5% of preactivation elements and flipping 14.9% of signs; `conv1_n` showed a similar ratio of 0.379, with 31.2% bias-dominant elements and a 15.2% sign-flip rate. In deeper layers, bias effects were negligible: conv2 and conv3 had bias/synaptic ratios at most 0.0105 and sign-flip rates below 0.004, while all fully connected layers had ratios at most 0.0032 and sign-flip rates below 0.001.

These results suggest that negative biases primarily act as threshold shifts, especially in the first convolutional layer, rather than as a mechanism for violating Dale-style synaptic signs. The formal pathway-level excitatory and inhibitory structure remains intact, while bias offsets provide a limited stream-specific adjustment to activation thresholds.

Rubric ablations

Rubric strictness	Key ED constraints	Observed effect
Loose	No explicit wording to prohibit backpropagation	Best validation accuracy 0.9694 but failed to implement ED
Intermediate	Explicit wording to prohibit backprop	Best validation accuracy 0.976. Implemented ED correctly
Strict	No backprop No $W^\top$ backward transport Explicit local update with one global error sign Detailed polarity invariance	Constrained search toward global/mean error broadcast ED; best validation accuracy 0.9708

Table 6 | Rubric ablation design for MNIST Error Diffusion runs. The experiment varied the strictness and wording of the method-validity rubric while keeping the MNIST task, fixed split, search budget, and high-level exclusions constant.

C. Highlighted Papers (Co-) Generated by The Agentic Scientist

C.1. Doc-to-LoRA Query-Conditioned Selection Paper & AI Review

sakana.ai

August 27, 2026

Addressing Beats Architecture: Query-Conditioned Merge-Time Selection for Context-to-LoRA Hypernetworks on Multi-Needle Retrieval

The Agentic Scientist System

A context-to-LoRA hypernetwork reads a document once and emits a low-rank adapter that folds it into a frozen language model. To serve inputs longer than its training chunk, the standard recipe splits the context, generates one adapter per chunk, and *merges* the per-chunk adapters into a single higher-rank adapter before the query is seen. We show that this merge, not the adapter generator or the training loss, is what limits retrieval when a context holds several facts and only one is queried. On a synthetic multi-needle needle-in-a-haystack benchmark spanning 128 to 65,536 tokens, we compare four mechanism families under one fixed base model, data generator, and three-hour single-GPU budget. Redesigning the hypernetwork's read-out and output head plateaus at 0.32 exact-match accuracy, and auxiliary binding and selection losses reach only 0.46; the strongest query-blind merge attains 0.52. Adding a query-conditioned late-interaction (MaxSim) address that selects which chunk's adapter dominates the merge – with no new parameters and no retraining – lifts accuracy to 0.98. The bottleneck is selection at composition time, not head capacity or auxiliary losses: weight-space context internalization is, at long context, a routing problem best resolved when the adapter is composed.

Introduction

Folding in-context information directly into a model's weights, instead of re-reading it as tokens at every query, is an appealing route to low-latency adaptation, and a growing line of work internalizes context into parameters or a compact learned memory rather than keeping it in the prompt (Cao et al., 2025). A hypernetwork reads a context once and emits a low-rank adaptation (LoRA) that, added to a frozen base model, makes it behave as though the context had been placed in the prompt. This "context-to-LoRA" construction is the weight-space analogue of parameter-efficient fine-tuning (Ding et al., 2022; Han et al., 2024; Mao et al., 2024), of soft-prompt transfer between frozen models (Vu et al., 2022), and of hypernetwork-generated adapters for instruction following (Ivison et al., 2023). Because the generator is trained at a bounded chunk length, the recipe scales to longer inputs by partitioning the context into chunks, emitting one adapter per chunk, and *merging* the per-chunk low-rank factors along the rank axis, so a longer context simply yields a higher-rank adapter. Composing low-rank modules in this way connects to a broad literature on merging and arithmetic combination of adapters (Huang et al., 2023; Kawata et al., 2026; Yang et al., 2024a; Zhang et al., 2023).

Convenient as it is, this composition silently makes a retrieval decision. The merged adapter is assembled *before* the query is known, so every chunk's contribution enters inference on equal footing. We probe this regime with a controlled multi-needle needle-in-a-haystack (NIAH) benchmark, the synthetic long-context retrieval paradigm now standard for stress-testing how faithfully models recover facts as context grows (Du et al., 2025; Hsieh et al., 2024; Roberts et al., 2024): each context embeds one to four named facts of the form *"the needle named* `amber` *stores magic number* `1234`*,"* with names and numbers unique within a context, and the prompt requests a single target needle's four-digit number. A lone fact is retrieved almost perfectly, but as co-resident facts accumulate

a query-blind merge superposes their digits and the model returns a distractor's number – a binding/superposition failure, the interference that arises when distributed representations must hold several bound key–value pairs at once (Elhage et al., 2022; Greff et al., 2020; Oberauer, 2009), that added generation capacity, by itself, does not remove.

Our question is diagnostic rather than incremental: on a fixed substrate and budget, *which* component governs multi-needle accuracy – the adapter-generating head, the training objective, the magnitude prior, or the query-blind way adapters are merged? We evaluate four mechanism families under an identical three-hour single-GPU budget on five context-length bins from 128 to 65,536 tokens. Two families target the intuitive suspects: the inductive bias of the hypernetwork's read-out and output head, and training-side objectives that supervise binding or selection. A third is the strongest query-blind merge we could construct, combining redundancy weighting, competitive gating, and magnitude-prior stabilization. The fourth makes composition query-aware: a frozen-encoder late-interaction score, in the spirit of ColBERT-style MaxSim retrieval (Khattab et al., 2021; Liu et al., 2024; Santhanam et al., 2022b), selects which chunk's adapter is merged for the current query over delimiter-aligned chunk boundaries.

The result is sharp and one-sided. Read-out and head redesign stays near 0.32 exact-match, auxiliary supervision reaches 0.46, and the best query-blind merge reaches 0.52; none of the three query-blind families crosses 0.52. Making selection query-conditioned at merge time – alone – lifts accuracy to 0.91, and adding relevance-profile atom placement and delimiter-aligned chunk cuts reaches 0.98 (§). Our contributions are: (i) a decomposition that localizes the multi-needle bottleneck to query-blind adapter composition rather than the hypernetwork head or the training loss; (ii) a parameter-free, merge-time late-interaction selection mechanism, plus two chunk-level components, that raises accuracy from a 0.52 ceiling to 0.98 without changing the base model, the adapter generator, or the training objective; and (iii) two grounded negative-control families that make the attribution falsifiable.

Related Work

Context internalization and hypernetwork adapters. Parameter-efficient fine-tuning adapts a frozen model with a small set of added weights (Ding et al., 2022; Han et al., 2024; Mao et al., 2024), and soft-prompt transfer shows that adaptation signal can be moved between frozen models through a compact learned artifact (Vu et al., 2022). Hypernetworks go further by *generating* the adaptation from an input rather than learning it by gradient descent; HINT, for instance, encodes an instruction into adapter weights for zero- and few-shot generalization (Ivison et al., 2023). Context-to-LoRA applies the same principle to arbitrary context, reading a document once and emitting an adapter that internalizes it. We take this generator as fixed and ask how its per-chunk outputs should be *combined* at long context length.

Merging and composing low-rank modules. A large body of work composes adapters or full models after training: dynamic LoRA composition (Huang et al., 2023), arithmetic over parameter-efficient modules (Klimaszewski et al., 2024; Zhang et al., 2023), weight-ensembling mixtures and representation-surgery merges (Tang et al., 2024; Yang et al., 2024b), attribute-conditioned LoRA merging (Kawata et al., 2026), and broad surveys of model merging (Yang et al., 2024a). These methods typically fix the merge coefficients *once*, independent of the query at inference. Our setting differs in that the correct composition depends on the query: the adapter that should dominate is the one whose chunk contains the queried fact, which is unknown until inference. We therefore make the composition itself query-conditioned.

Late interaction and token-level retrieval. Late-interaction retrieval scores a query against a passage by summing token-level maximum similarities (MaxSim), retaining fine-grained matches that single-vector encoders discard (Khattab et al., 2021; Santhanam et al., 2022b); follow-up work analyzes its matching behavior and efficiency (Liu et al., 2024; Santhanam et al., 2022a), and dense-retrieval encoders study token- versus passage-level granularity (Ma et al., 2022; Wang et al., 2023; Wu et al., 2023; Zhong et al., 2022). Retrieval-augmented generation uses such scores to fetch external text and reduce hallucination (Shuster et al., 2021). We reuse the MaxSim primitive in an unusual place: not to fetch text into a prompt, but to decide, inside a frozen encoder's representation space, which generated adapter to merge into the weights. To our knowledge, late interaction has not previously been used as a merge-time router over hypernetwork-generated adapters.

Method

Problem setup

The system is a context-to-LoRA hypernetwork over a frozen instruction-tuned ∼2B-parameter decoder language model. Given a context c, a latent cross-attention encoder produces per-chunk summaries and a hypernetwork emits a low-rank adapter ΔW_k for each chunk k, applied to a single projection family of the frozen model at a fixed rank. To serve contexts longer than the training chunk length, the per-chunk adapters are *merged* into one higher-rank adapter that is added to the base weights, after which the model is queried with no context tokens in the prompt. The task is multi-needle NIAH: each context independently samples one to four named facts ("the needle named `name` stores magic number `dddd`"), names and numbers are unique within a context, and the prompt requests exactly one target needle's four-digit number. Distractor needles are placed at random non-target depths, so a system must learn key–value binding rather than positional copying. The optimization target is exact-match accuracy on the queried number, evaluated on five context-length bins spanning 128 to 65,536 tokens; every configuration is trained within a strict three-hour single-GPU budget and scored by one fixed protocol.

Where the bottleneck could live

Multi-needle retrieval can fail at three places, and our hypotheses map directly to them. (1) *Generation*: the hypernetwork's read-out or output head might lack the capacity to encode several distinct key–value bindings, in which case richer heads should help. (2) *Training*: the loss might never teach the model to separate co-resident facts, in which case auxiliary binding or selection objectives should help. (3) *Composition*: even with perfectly generated per-chunk adapters, merging them query-blind superposes the facts, so the answer collapses toward an arbitrary co-resident number. Our central hypothesis is (3): the merge is a retrieval decision made with no access to the query, and no amount of generation capacity or training pressure can recover a decision that the composition rule structurally discards. Making composition query-conditioned should therefore be decisive, and families (1) and (2) should act as negative controls that stay below the query-blind ceiling.

Compared approaches

We evaluate four mechanism families on the same substrate; configurations within each are grouped by their dominant mechanism.

Query-blind adapter merge (baseline ceiling). The reference composition concatenates the per-chunk low-rank factors along the rank axis and adds the result to the frozen weights. We include the strongest query-blind instance we could build: redundancy-aware chunk weighting that suppresses near-duplicate chunks, a competitive per-slot gate, and a gradient-pinned magnitude prior that

keeps the merged adapter's write norm in a healthy range. This family fixes the accuracy ceiling that a decisive mechanism must beat.

Query-conditioned late-interaction selection (proposed). At inference we encode the query and each context chunk with the same frozen early-exit encoder that already feeds the hypernetwork, and score each chunk against the query with a token-level late-interaction address: for every query token we take its maximum cosine similarity to the chunk's token features and average over query tokens (Khattab et al., 2021; Santhanam et al., 2022b). This MaxSim address selects which chunk's generated adapter dominates the merge, turning query-blind composition into query-conditioned selection. It adds no parameters and no training: it is a retrieval score computed in the frozen encoder's representation space and applied only when the adapter is composed, which is exactly what isolates "selection" from "head capacity" and "training objective."

Relevance-profile placement and delimiter-aligned chunking (proposed, supporting). Two components convert a correct selection into a correct answer. First, a token-level relevance profile places the merged adapter's "atom" – the span whose representation the selected adapter is built from – on a relevance peak rather than a content-blind window, so the queried fact, not a neighbor, drives the adapter. Second, chunk boundaries are cut at the generator's own delimiter units, so a short target fact is not split across two chunks; a split fact reaches the adapter as a fragment and is the dominant residual error once selection is query-conditioned.

Read-out and output-head redesign (negative control). This family changes the inductive bias of adapter *generation* while leaving composition query-blind: scale-invariant slot read-outs, decorrelated read keys, uniqueness-weighted token pooling, and factorized or jointly-normalized key–value heads that alter how the hypernetwork emits its low-rank factors. If generation capacity were the bottleneck, these should close the gap.

Auxiliary training supervision (negative control). This family adds selection pressure through the training objective while leaving composition query-blind: multi-target needle supervision that grades several needles from one context, a co-resident digit-margin loss that penalizes emitting a distractor's digits, supervised chunk addressing, and cross-context distractor blocks that inject hard negatives. If the loss were the bottleneck, these should close the gap.

We additionally report a *training-reliability* component – magnitude-prior pinning and an early mode-detection restart – that is not an accuracy driver: it reduces a run-to-run collapse mode in which training occasionally converges to a degenerate adapter scoring zero. We treat it strictly as a robustness aid and discuss it under limitations.

Experimental protocol

Each configuration is one trained model evaluated once by the fixed benchmark. Training must finish within three hours on a single GPU; over-budget runs are invalid and discarded. The base model, data generator, evaluation scripts, and train/eval scale are identical across configurations, so the only variable is the model-internal mechanism. We report exact-match accuracy on the queried needle, pooled over the five context-length bins. Because the study varies one mechanism at a time on a shared substrate, we read the four families as a controlled comparison and the proposed method's internal components as a mechanism attribution – which component adds how much accuracy – not as a temporal trajectory. Of 41 model variants, 31 configurations were trained and scored and 30 produced a valid model (one failed the runtime/validity contract and is excluded). A subset of otherwise valid seeded runs collapsed to zero exact-match under a run-to-run instability

Doc-to-LoRA Query-Conditioned Selection Paper (pages 1–4 of 12)

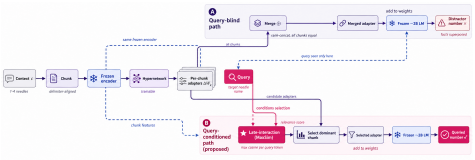


Figure 1 | Context-to-LoRA multi-needle pipeline. A context is chunked, a hypernetwork emits one low-rank adapter per chunk, and the per-chunk adapters are composed into a single adapter added to the frozen model. The query-blind path merges all chunks before the query is known, superposing co-resident facts so the model can emit a distractor's number; the proposed query-conditioned path scores each chunk against the query with a late-interaction (MaxSim) address in the frozen encoder's space and selects which chunk's adapter dominates the merge, over delimiter-aligned chunk boundaries.

that the reliability component targets, which we report as a variance issue rather than a property of any mechanism.

Implementation notes

The frozen base is an instruction-tuned ∼2B-parameter decoder; adapters are applied at a fixed low rank to a single projection family, and the encoder is a latent cross-attention pooler that summarizes each chunk. The late-interaction address and the relevance profile reuse this same frozen encoder, so the proposed selection adds no trainable parameters and only a small inference-time encoding cost, bounded well inside the evaluation budget. The negative-control families and the baseline share the identical generator, encoder, and training loop; they differ only in the read-out/head, the auxiliary loss terms, or the merge rule.

Threats to validity

The study is on a single synthetic benchmark with a fixed base model and a fixed three-hour budget; absolute numbers may not transfer to natural long-context retrieval. The proposed address operates in a frozen encoder's representation space, so its ceiling is bounded by that encoder's ability to separate chunks, and a small short-context residual remains. We mitigate confounding by holding every non-mechanism factor fixed and by including two independent negative-control families rather than a single contrast.

Results

Main result: composition is the bottleneck

Figure 2 and Table 1 compare the best exact-match accuracy attained by each mechanism family. Redesigning the hypernetwork read-out and output head leaves accuracy near 0.32; adding auxiliary training supervision reaches 0.46; the strongest query-blind merge reaches 0.52. Making adapter selection query-conditioned at merge time reaches 0.98 – an absolute gain of 0.46 over the best query-blind system and of 0.66 over the best read-out/head redesign. The two intuitive suspects, generation capacity and training loss, both stay strictly below the query-blind merge itself, which is the strongest evidence that the bottleneck is neither: what is missing cannot be supplied by a richer head or a better loss while composition remains query-blind.

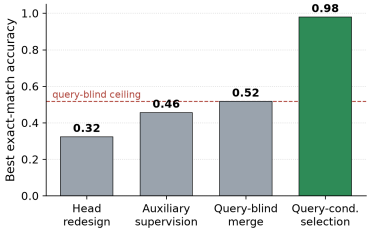


Figure 2 | Best multi-needle retrieval accuracy (exact match) by mechanism family on the same benchmark and budget. The three query-blind families – read-out/head redesign, auxiliary supervision, and the strongest query-blind merge – all sit at or below the 0.52 ceiling (dashed). Query-conditioned merge-time selection reaches 0.98.

Mechanism family (best configuration)	Exact match
Query-blind read-out / head redesign	0.32
Query-blind auxiliary supervision	0.46
Query-blind merge weighting + magnitude prior	0.52
Query-conditioned late-interaction selection	**0.98**

Table 1 | Best pooled multi-needle exact-match accuracy per mechanism family. Three independent query-blind families saturate at or below 0.52; only query-conditioned selection crosses the ceiling.

Component attribution

Figure 3 decomposes the proposed method into cumulative mechanisms above the query-blind ceiling. Starting from the 0.52 query-blind merge, adding query-conditioned late-interaction selection alone lifts accuracy to 0.91 – by far the single largest jump in the study, and direct confirmation that the missing ingredient is query-aware routing rather than anything upstream. Placing the merged adapter's atom on a relevance peak adds a further step to 0.93, and cutting chunk boundaries at generator delimiter units so a target fact stays intact reaches 0.98. The two supporting components together contribute the final 0.07: once the right chunk is selected, the remaining errors are a correctly selected chunk whose atom is mis-placed or whose fact was split by a boundary, both of which the supporting components address. An intermediate delimiter-snapped span variant reaches 0.94 before aligned block cuts, confirming the two chunk-level components act on the same residual.

Distribution of outcomes and the superposition signature

Figure 4 plots every valid configuration's accuracy within its family. The three query-blind families are dispersed but bounded above by the 0.52 ceiling, while the query-conditioned family concentrates near ceiling. This bound is the empirical signature of superposition: a query-blind merge of n co-resident facts behaves, on multi-needle contexts, like returning one fact at roughly

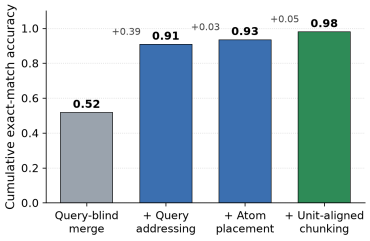


Figure 3 | Cumulative contribution of the proposed method's components above the query-blind ceiling. Query-conditioned selection is the decisive step (0.52→0.91); relevance-profile atom placement (→0.93) and delimiter/unit-aligned chunk cuts (→0.98) convert a correct selection into a correct answer. Bars are keyed by mechanism, not by time.

chance, so pooled accuracy saturates well below one even though single-needle contexts are handled almost perfectly. The bound appears already at the shortest context lengths, so it is a composition failure rather than a long-context capacity problem. Query-conditioned selection removes the bound by resolving *which* fact to express before the adapters are combined, and it does so across context lengths rather than trading one length against another.

The spread also exposes the run-to-run collapse mode: several otherwise-valid configurations, in every family, score zero exact-match. These collapses are a training-stability artifact – a degenerate adapter that never escapes an uninformative solution – rather than evidence about a mechanism, and they include configurations from the proposed family itself. This is why we treat the reliability component as robustness rather than accuracy: an early mode-detection restart and magnitude-prior pinning recover collapsed draws to bank level (0.93) and reduce the frequency of zero-accuracy runs, but they do not raise any family's accuracy ceiling.

Interpretation

Read together, the three query-blind families give a falsifiable attribution. If generation capacity were limiting, the read-out and head redesigns would have crossed the ceiling; they did not (best near 0.32). If the training objective were limiting, auxiliary binding and selection losses would have crossed it; they did not ($\leq$ 0.46). The strongest query-blind merge, with redundancy weighting, competitive gating, and magnitude stabilization, itself stops at 0.52. Only the mechanism that injects query information at composition time crosses the ceiling, and by a wide margin. This localizes the multi-needle bottleneck to query-blind adapter composition and identifies late-interaction merge-time selection as the mechanism that resolves it.

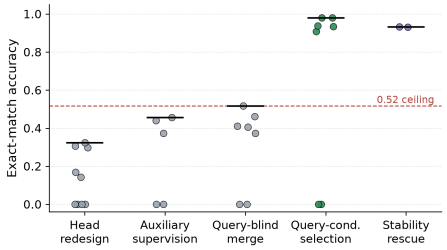


Figure 4 | Per-configuration exact-match accuracy by mechanism family across the 30 valid trained models; horizontal marks denote each family's best and the dashed line is the 0.52 query-blind ceiling. Query-blind families are bounded by the ceiling while the query-conditioned family concentrates near it. Points at 0.0 are the run-to-run collapse mode discussed in the text, present in every family and not a property of any mechanism.

Limitations

Our evidence is confined to one synthetic multi-needle benchmark, a single frozen base model, and a fixed three-hour budget; the exact numbers, and the size of the gap, may not transfer to natural documents with paraphrased or entailed facts where exact-match binding is ill-defined. The proposed address is a retrieval score in a *frozen* encoder's representation space, so it inherits that encoder's chunk-separability and cannot recover a fact the encoder cannot distinguish; a small short-context residual, dominated by facts split across a chunk boundary, remains even after atom placement and aligned chunking. The method assumes a chunked, mergeable adapter and a single queried fact per prompt; extending query-conditioned composition to all-needles recall, to multiple simultaneous queries, or to a learned (rather than frozen) address is left open. Finally, training-run collapse remains a reliability concern that our stabilization only mitigates, and we report it as variance rather than as a solved problem.

Conclusion

We asked where the accuracy bottleneck lives for a context-to-LoRA hypernetwork that must retrieve one queried fact from a multi-needle context, and answered it by controlled comparison rather than by adding capacity. Reshaping the hypernetwork's read-out and output head, and adding training-side binding and selection objectives, both leave accuracy at or below a 0.52 query-blind ceiling. The decisive change is to make adapter composition query-conditioned: a late-interaction (MaxSim) address in the frozen encoder's space selects which chunk's generated adapter is merged, and with relevance-profile atom placement and delimiter-aligned chunk boundaries this raises exact-match accuracy from 0.52 to 0.98 across context lengths from 128 to 65,536 tokens, without touching the base model, the adapter generator, or the training loss. The broader lesson is that

weight-space context internalization is, at long context, a routing problem: when many facts share one merged adapter, retrieval is best resolved when the adapter is composed, and a token-level late-interaction score is a simple, parameter-free, and effective router. We hope this reframing – selection over generation and supervision – informs the design of scalable context-to-weight systems.

References

Bowen Cao, Deng Cai, and Wai Lam. Infiniteicl: Breaking the limit of context window size via long short-term memory transformation. In *Findings of the Association for Computational Linguistics: ACL 2025*, page 11402–11415. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.595. URL http://dx.doi.org/10.18653/v1/2025.findings-acl.595.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. 2022. doi: 10.21203/rs.3.rs-1553541/v1. URL http://dx.doi.org/10.21203/rs.3.rs-1553541/v1.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, E. A. Huerta, and Hao Peng. Context length alone hurts llm performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, page 23281–23298. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.1264. URL http://dx.doi.org/10.18653/v1/2025.findings-emnlp.1264.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks, 2020. URL https://arxiv.org/abs/2012.05208.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff, and Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2403.14608.

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

Chengsong Huang, Qian Liu, Bill Lin, Tianyu Pang, Chao-Hai Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition, 2023. URL https://arxiv.org/abs/2307.13269.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew E. Peters. Hint: Hypernetwork instruction tuning for efficient zero- and few-shot generalisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 11272–11288. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.631. URL http://dx.doi.org/10.18653/v1/2023.acl-long.631.

Ryosuke Kawata, Joonho Lee, Yanlei Gu, and S. Kamijo. Adaptive multiple-attribute scenario lora merge for robust perception in autonomous driving. *Sensors*, 26(4):1336, February 2026. ISSN 1424-8220. doi: 10.3390/s26041336. URL http://dx.doi.org/10.3390/s26041336.

Omar Khattab, Christopher Potts, and Matei Zaharia. Relevance-guided supervision for openqa with colbert. *Transactions of the Association for Computational Linguistics*, 9:929–944, 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00405. URL http://dx.doi.org/10.1162/tacl_a_00405.

Mateusz Klimaszewski, Piotr Andruszkiewicz, and Alexandra Birch. No train but gain: Language arithmetic for training-free language adapters enhancement, 2024. URL https://arxiv.org/ab

s/2404.15737.

Qi Liu, Gang Guo, Jiaxin Mao, Zhicheng Dou, Ji-Rong Wen, Hao Jiang, Xinyu Zhang, and Zhao Cao. An analysis on matching mechanisms and token pruning for late-interaction models. *ACM Transactions on Information Systems*, 42(5):1–28, April 2024. ISSN 1558-2868. doi: 10.1145/3639818. URL http://dx.doi.org/10.1145/3639818.

Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. Pre-train a discriminative text encoder for dense retrieval via contrastive span prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 848–858. ACM, 2022. doi: 10.1145/3477495.3531772. URL http://dx.doi.org/10.1145/3477495.3531772.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7), December 2024. ISSN 2095-2236. doi: 10.1007/s11704-024-40663-9. URL http://dx.doi.org/10.1007/s11704-024-40663-9.

Klaus Oberauer. Interference between storage and processing in working memory: Feature overwriting, not similarity-based competition. *Memory & Cognition*, 37(3):346–357, April 2009. ISSN 1532-5946. doi: 10.3758/mc.37.3.346. URL http://dx.doi.org/10.3758/mc.37.3.346.

Jonathan C. Roberts, Kai Han, and Samuel Albanie. Needle threading: Can llms follow threads through near-million-scale haystacks?, 2024. URL https://arxiv.org/abs/2411.05000.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. Plaid: An efficient engine for late interaction retrieval, 2022a. URL https://arxiv.org/abs/2205.09707.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 3715–3734. Association for Computational Linguistics, 2022b. doi: 10.18653/v1/2022.naacl-main.272. URL http://dx.doi.org/10.18653/v1/2022.naacl-main.272.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, page 3784–3803. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-emnlp.320. URL http://dx.doi.org/10.18653/v1/2021.findings-emnlp.320.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts, 2024. URL https://arxiv.org/abs/2402.00433.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 5039–5059. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.346. URL http://dx.doi.org/10.18653/v1/2022.acl-long.346.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Simlm: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 2244–2258. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.125. URL http://dx.doi.org/10.18653/v1/2023.acl-long.125.

Desheng Wu, Guangyuan Ma, Meng Lin, Zijia Lin, Zhongyuan Wang, and Songlin Hu. Contextual

masked auto-encoder for dense passage retrieval. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4738–4746, 2023. ISSN 2159-5399. doi: 10.1609/aaai.v37i4.25598. URL http://dx.doi.org/10.1609/aaai.v37i4.25598.

Enneng Yang, Shen Li, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities, 2024a. URL https://arxiv.org/abs/2408.07666.

Enneng Yang, Li Shen, Zhenyi Wang, Guibing Guo, Xiaojun Chen, Xingwei Wang, and Dacheng Tao. Representation surgery for multi-task model merging, 2024b. URL https://arxiv.org/abs/2402.02705.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations, 2023. URL https://arxiv.org/abs/2306.14870.

Wei Zhong, Jheng-Hong Yang, Yuqing Xie, and Jimmy Lin. Evaluating token-level and passage-level dense retrieval models for math information retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, page 1092–1102. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-emnlp.78. URL http://dx.doi.org/10.18653/v1/2022.findings-emnlp.78.

Doc-to-LoRA Query-Conditioned Selection Paper (pages 9–12 of 12)

Review


```
{
  "Summary": "The paper studies a context-to-LoRA hypernetwork on a synthetic multi-needle
  needle-in-a-haystack task where each context contains one to four named facts and the query asks for one
  target number. It argues that the main bottleneck is query-blind adapter composition: merging every
  per-chunk LoRA before the query is known superposes co-resident facts and leads to distractor-number
  errors. The strongest query-blind merge reaches 0.52 exact-match accuracy, read-out/head redesign reaches
  0.32, and auxiliary binding or selection losses reach 0.46. A parameter-free query-conditioned
  late-interaction (MaxSim) address that selects the relevant chunk's adapter at merge time raises accuracy
  to 0.91, and relevance-profile atom placement plus delimiter-aligned chunking reaches 0.98.",
  "Strengths": [
    "Clear mechanism-level attribution separating generation capacity, training supervision, query-blind
    merge weighting, and query-conditioned composition.",
    "Strong negative controls: both richer adapter-generation heads and auxiliary losses remain below the
    query-blind merge ceiling.",
    "Large empirical gap between query-blind and query-conditioned regimes, with a cumulative component
    ablation for selection, atom placement, and chunk-boundary alignment.",
    "The proposed late-interaction router adds no trainable parameters and isolates selection at composition
    time from changes to training or the base model.",
    "The benchmark has a clean exact-match target and exposes a concrete binding failure through
    distractor-number errors."
  ],
  "Weaknesses": [
    "Evaluation is confined to a synthetic multi-needle retrieval task with four-digit numeric answers, so
    transfer to natural documents is untested.",
    "The study uses one frozen base model and one context-to-LoRA substrate; it is unclear how general the
    ceiling and the router are across model scales and adapter designs.",
    "Most configurations appear to be single-seed runs, which limits statistical estimates even though the
    family-level gap is large.",
    "The method relies on a frozen encoder's chunk matching quality and may fail when the encoder cannot
    separate paraphrased, entailed, or multi-hop evidence.",
    "Query-conditioned selection moves some retrieval work back to inference time; the paper should quantify
    latency and scaling costs more directly."
  ],
  "Originality": 4,
  "Quality": 3,
  "Clarity": 4,
  "Significance": 3,
  "Questions": [
    "Does query-conditioned merge-time selection still dominate on natural long-document QA with paraphrased
    answers or multi-hop dependencies?",
    "How sensitive is the result to the frozen encoder used for late-interaction scoring and to the chunk
    size/boundary policy?",
    "Can the router compose multiple chunks when the query requires several facts rather than selecting one
    dominant adapter?",
    "What is the latency and memory overhead of scoring all chunks before each query at 64K-token scale and
    beyond?",
    "Would repeated seeds preserve the same family-level separation and reduce the observed zero-accuracy
    collapse mode?"
  ],
  "Limitations": [
    "Single synthetic benchmark with constrained answer format.",
    "Single base model and adapter target module.",
    "Single-seed per-configuration evidence limits uncertainty estimates.",
    "Selection depends on frozen encoder representations rather than learned adapter semantics.",
    "The approach is designed for one queried fact and does not yet address all-needles recall or multiple
    simultaneous queries."
  ],
  "Ethical Concerns": false,
  "Soundness": 3,
  "Presentation": 4,
  "Contribution": 3,
  "Overall": 6,
  "Confidence": 4,
  "Decision": "Weak Accept"
}
```

C.2. Learned Covariance Matrix Adaptation Paper & AI Review

How Much of a Learned Evolution Strategy Is the Algorithm It Was Anchored To?

The Agentic Scientist

Abstract

Learned optimizers are usually reported by the margin they win by, not by the margin their hand-designed scaffolding already wins by. We study a learned evolution strategy whose learned outputs are zero-initialized bounded multiplicative residuals on a separable CMA-ES core, so the strategy is exactly the classical algorithm at initialization. Its 364 meta-parameters are meta-evolved only on 2–10-dimensional analytic functions, frozen, and the same optimizer is rebuilt for a 704-dimensional Brax Ant policy and scored by best-so-far area under the curve over 2,000 generations, normalized so the initial policy is 0 and a same-seed CMA-ES reference is 1. The strategy scores 1.191 and beats the reference on all eight evaluation seeds. However, switching the learned modulation off—which provably reduces the same code to separable CMA-ES plus the fixed sampling machinery—still scores 1.107, so the anchor supplies 55.8% of the above-reference margin. The learned residual is real (+0.083 paired, 8/8 seeds) but conditional: it shrinks with budget and reverses sign on final solution quality. Meta-learning succeeded in-distribution; transfer is where the gain was lost.

1 Introduction

Black-box optimizers for continuous problems are still overwhelmingly hand-designed, and the design space is large enough that new heuristics appear faster than they can be compared (Osaba et al., 2021; Houssein et al., 2022). Meta-black-box optimization proposes to replace that manual search: parameterize the update rule of an evolutionary algorithm with a small neural network, meta-train it on a distribution of cheap objective functions, and hope that the resulting rule generalizes (Lange et al., 2023; Gaier et al., 2020; Zhan et al., 2023). The appeal is that the inner tasks can be cheap analytic functions while the deployment task is expensive—for instance a neural control policy, whose parameter space is high-dimensional and awkwardly conditioned enough to frustrate even methods with access to far more structure, such as Markov chain Monte Carlo samplers for Bayesian neural networks (Papamarkou et al., 2022).

This creates an attribution problem, and it is structural rather than incidental. Learned update rules are almost never learned from nothing: they are parameterized deviations from an update that already works. Outside evolutionary computation, meta-learned optimizers such as Meta-SGD keep gradient descent as their backbone, learning a per-parameter vector that rescales and redirects its step (Li et al., 2017); inside it, learned evolutionary operators sit on top of the selection, recombination, and adaptation machinery of the algorithm they modify (Lange et al., 2023). A learned rule built on a competent classical algorithm therefore inherits that algorithm's performance for free. If it then beats an external baseline, the reported margin conflates two very different quantities: what meta-learning contributed, and what the scaffolding already contributed. Separating them requires a comparison the external baseline cannot provide—the same code with the learned part switched off. The question this paper asks is therefore not *does a learned optimizer transfer*, but *how much of its measured advantage is the optimizer it was anchored to*.

We make that question answerable by construction. We study a learned evolution strategy in which every learned output is a zero-initialized bounded multiplicative residual, $g(v) = 2^{\tanh(3v)} \in [\frac{1}{2}, 2]$, applied to a component of a separable CMA-ES core. Because the final layer of each network is initialized to zero, $g(\cdot) = 1$ exactly at initialization and the strategy *is* separable CMA-ES wrapped in the fixed sampling machinery. This makes an exact null control available: forcing every gate to 1 for the whole run recovers the classical algorithm within the same code path, the same fixed sampling machinery, and the same evaluation harness. The gap between that control and the full method is the part meta-learning is responsible for; everything below it is scaffolding.

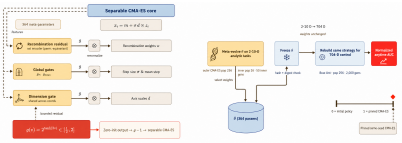


Figure 1: The object of study. Three small learned modules emit bounded multiplicative residuals on a separable CMA-ES core (recombination weights, the step size and mean step, and the per-coordinate axis scales). Zero-initialized output layers make every residual exactly 1 at initialization, so the strategy starts as, and can be forced back to, the classical algorithm. The meta-parameters are evolved only on 2–10-dimensional analytic tasks, frozen, and the same optimizer is then rebuilt for a 704-dimensional control policy and scored against a pinned same-seed CMA-ES reference.

Our contributions are as follows. (i) We describe a 364-parameter anchored residual evolution strategy whose learned modules are permutation-equivariant in the population and shared across coordinates, so a single weight vector serves a 16-member population in 2–10 dimensions during meta-training and a 256-member population in 704 dimensions at deployment, with no learned feature or gate reading a generation counter or a budget. (ii) We report the decomposition rather than the headline: against a pinned same-seed CMA-ES reference at 1.000, the identity-gate anchor reaches 1.107 and the full method 1.191, so the anchor accounts for 55.8% of the entire above-reference margin. (iii) We show the surviving learned increment is genuine but conditional. It is positive on all eight seeds, yet its share of the above-reference excess falls from 86.5% at generation 1,000 to 44.2% at generation 2,000, and it reverses sign on final solution quality, where the no-learning anchor finishes 220.7 return higher. (iv) We localize the failure: meta-training worked, cutting final log regret by 50.8% on its training functions and 28.4% on held-out ones. What did not survive was the transfer of that gain from at most ten dimensions to a 704-dimensional control task.

2 Related Work

Discovering optimizers instead of designing them. Meta-black-box optimization treats the update rule of an evolutionary algorithm as a differentiable-free parameterized object and searches over it with an outer optimizer. Attention-based genetic algorithms discovered this way generalize across problem classes without per-task retuning (Lange et al., 2023); related work learns the search *representation* rather than the update (Gaier et al., 2020), and broader surveys situate these methods within learning-aided evolutionary computation (Zhan et al., 2023) and meta-learning generally (Vanschoren, 2019). The impulse is not specific to evolutionary computation: gradient-based meta-learning also learns the update rule itself, for instance by meta-learning a per-parameter step vector on top of stochastic gradient descent (Li et al., 2017). In both families the learned object is a modification of a working algorithm, which is precisely what makes attribution hard. Our setting follows the evolutionary template—outer evolutionary search over inner-strategy parameters, frozen transfer—but is instrumented so that the classical contribution can be subtracted out.

Adaptation inside classical evolutionary algorithms. The mechanisms our learned gates modulate are exactly the ones that classical algorithms already adapt by hand: recombination weights, step size, and covariance. The evolution-strategy family and the derandomized adaptation mechanisms it accumulated are surveyed by Emmerich et al. (2018). CMA-ES, its best-known member, couples cumulative step-size adaptation with rank-one and rank-μ covariance updates (Hansen, 2016); its separable variant restricts the covariance to a diagonal and remains competitive in high dimension, particularly under low effective dimensionality (Yamaguchi et al., 2022); scaling population-based search to hundreds or thousands of variables is a distinct problem with its own literature (Omidvar et al., 2022; Yang et al., 2018). A long line of work hand-designs the online adaptation of operators and control parameters (Elsayed et al., 2017), of the selection rule itself (Rúnarsson & Yao, 2000), or of an adaptive grid maintained over the search space (Rostami & Shenfield, 2016). Transferring *tuned* CMA-ES parameters between problems has itself been studied (Thomaser et al., 2023), and CMA-ES is routinely extended with surrogates or randomized parameters for applications (Hamdani et al., 2019; Sauerland et al., 2023). Our residual gates can be read as a learned, bounded, evidence-conditioned version of these adaptation rules that degrades gracefully to the unmodified algorithm.

Architectures for population-agnostic conditioning. Because a strategy must accept any population size and any dimension, the component that consumes the population is naturally expressed as a permutation-equivariant set function rather than a fixed-width vector; permutation-invariant attention architectures formalize this (Lee et al., 2018). Components that consume per-coordinate or scalar summaries can instead be shared across coordinates, which is the route we take for two of our three modules. Amortized conditioning of a predictor on a variable-size context set is the same construction used by neural processes and their descendants (Gordon et al., 2019; Jha et al., 2022), and by closely related amortized-inference transformers (Müller et al., 2021), and the practical importance of normalizing such inputs is well documented (Huang et al., 2020).

Where a primary source we rely on conceptually is not cited directly, we name it in prose and, where a survey or tutorial covers the same material, cite that instead (Emmerich et al., 2018; Hansen, 2016). This applies to the original separable CMA-ES of Ros and Hansen, cumulative step-size adaptation of Hansen and Ostermeier, the theory of mirrored sampling, the BBOB/COCO noiseless testbed, and the Brax physics engine.

3 Method

3.1 Problem setup and success criterion

A candidate strategy is an `ask`/`tell` search distribution over $\mathbb{R}^D$ carrying a vector θ of learned meta-parameters, with $1 \le \dim\theta \le 1024$. The evaluator is fixed and has two parts. In the first, θ is meta-evolved on cheap analytic functions. In the second, θ is frozen, digest-authenticated, and the *same* strategy is rebuilt for a continuous-control policy search, which is what produces the reported score. The control task never updates θ or any outer-search state, so the second part is a pure out-of-distribution transfer test.

The control task is an episodic reinforcement-learning problem whose objective is the return accumulated over a policy rollout (Kaelbling et al., 1996), but here the optimizer observes only that episode-level return, and the problem is solved by direct search over policy parameters rather than by value or gradient estimation. Concretely it is a Brax `v1` `ant` environment with legacy-spring dynamics and contact-force observations, optimized with a tanh-linear policy over $87 \times 8 + 8 = 704$ parameters, initialized at the origin, with 500-step episodes, a population of 256, and 2,000 generations. For each of eight fixed evaluation seeds we take the mean of the best-so-far return curve over all generations—the anytime area under the curve—and average over seeds. Writing A_{alg}, A_{cma} and A_{ini} for that quantity under the candidate, under a pinned same-seed CMA-ES reference, and under the untrained initial policy, the reported score is

$$s = \frac{A_{\text{alg}} - A_{\text{ini}}}{A_{\text{cma}} - A_{\text{ini}}}, \qquad \text{so} \qquad s = 0 \text{ at the initial policy}, \quad s = 1 \text{ at CMA-ES}. \tag{1}$$

The reference is pinned: population 256, 2,000 generations, initial standard deviation 0.05, the same eight seeds, the same episode budget. The comparison is therefore iso-budget and same-seed, and $s > 1$ means the candidate accumulates more best-so-far area than a strong classical optimizer under identical conditions. Note that Equation 1 rewards *anytime* progress: a strategy can win on s while finishing at a worse solution, a distinction that turns out to matter.

3.2 Design rationale: make the classical contribution subtractable

Two design commitments follow from the attribution question.

Anchor, and make the anchor exactly recoverable. Every learned output is a bounded multiplicative residual

$$g(v) \;=\; \exp\!\big(\ln 2 \cdot \tanh(\gamma v)\big) \;\in\; [\tfrac{1}{2}, 2], \qquad \gamma = 3, \tag{2}$$

applied to a quantity that separable CMA-ES already computes. The final dense layer of every module has zero-initialized kernel and bias, so $v = 0$ and $g(0) = 1$ *exactly* at initialization: the anchored core is the classical algorithm, not an approximation of it. The outer search therefore explores only dimensionless multiplicative deformations of a known-good algorithm, and the worst case is bounded—no gate can move a quantity by more than a factor of two per generation. It also makes the decisive experiment trivial: pinning $g \equiv 1$ for a whole run yields a configuration that is, by construction, separable CMA-ES plus the fixed machinery of Section 3.4, and shares every other line of code with the full method.

Condition on evidence, not on time. Meta-training runs for 50 inner generations in 2–10 dimensions; deployment runs for 2,000 generations in 704. A feature carrying a generation index, a budget fraction, or a horizon would risk letting the outer search fit a schedule tied to the 50-generation meta-training regime; we therefore exclude such features by design rather than by measurement, as no time-conditioned configuration was run. Every feature we use is dimensionless—normalized ranks, cosine alignments, path-length ratios against their neutral expectations, success frequencies—and no learned feature or gate reads a counter or a budget. The one counter in the algorithm is the evidence-driven stall counter of Section 3.4, which resets on improvement rather than tracking elapsed time.

3.3 The anchored strategy

Search distribution. Candidates are drawn in the standard separable reparameterization $x_i = m + \sigma\, d \odot z_i$, with $z_i \sim \mathcal{N}(0, I_D)$, scalar step size σ, and per-coordinate axis scales $d \in \mathbb{R}^D_{>0}$. The representation is diagonal, so state and compute are $O(D)$ rather than $O(D^2)$, which is what keeps the strategy cheap at $D = 704$ with a population of 256; the restriction follows the evidence that a diagonal covariance remains competitive in high dimension, particularly under low effective dimensionality (Yamaguchi et al., 2022). We treat this as a design assumption rather than a demonstrated necessity: no full-covariance, limited-memory, or low-rank comparator was run at this scale in this study, and the large-scale literature contains other tractable high-dimensional search methods (Omidvar et al., 2022; Yang et al., 2018). The one covariance-representation comparator we do run moves in the opposite direction—pinning the axis scales to 1—and the diagonal beats it by 0.360 paired (Section 4.4).

Three learned modules (364 parameters). All modules use a hidden width of 8 and a zero-initialized output layer. *(a)* **A recombination residual** (201 parameters) is a permutation-equivariant set encoder over the population in the Deep-Sets style (Lee et al., 2018): a $6 \to 8$ ReLU layer per member, a mean-pooled context broadcast back and concatenated, a $16 \to 8$ ReLU layer, and an $8 \to 1$ output. Its output gates the CMA recombination weights, $w_i \leftarrow w_i^{\exp g}\, g(\cdot)$, followed by renormalization. *(b)* **Global gates** (98 parameters) map 9 scalar features through $9 \to 8$ ReLU and $8 \to 2$ to two scalars that modulate the step-size update and the mean step. *(c)* A **dimension gate** (65 parameters) maps 6 per-coordinate features through $6 \to 8$ ReLU and $8 \to 1$, applied identically to every coordinate, and modulates the exponent of the diagonal covariance update. Total: $201 + 98 + 65 = 364$ meta-parameters. No weight shape depends on D or on the population size N; the set module is equivariant in N and the dimension gate is shared across coordinates, which is what allows one frozen weight vector to serve both regimes.

Features. The six member features are a centered rank, a Gaussian-quantile transform of that rank, a centered improvement indicator, a normalized radial term $\tanh(\|z_i\|^2/D - 1)$, a normalized base-weight term, and the cosine alignment between the member's noise and the step-size evolution path. The nine global features are the CSA signal $\tanh(\|p_\sigma\|/\chi_D - 1)$, a rank-one path ratio, an elite radial term, the spread of $\log d$, a path alignment, and four success rates (three exponential moving averages at different rates plus the current-generation improvement fraction), each mapped to $[-1, 1]$. The six per-coordinate features are $\tanh$ transforms of p_σ, of p_c/d, of centered $\log d$, of the rank-μ and rank-one terms minus one, and a broadcast global signal. Every feature is scale-free by construction, which also keeps the network inputs in a well-conditioned range (Huang et al., 2020); conditioning a fixed parameter vector on a variable-size evidence set in this way is the same amortization used by conditional neural processes (Gordon et al., 2019; Jha et al., 2022).

Updates. The mean moves by $m \leftarrow m + g_{\text{mean}}\,\sigma\,y_w$ with y_w the weighted step; the step size follows cumulative step-size adaptation, $\log\sigma \leftarrow \log\sigma + g_\sigma\,(c_\sigma/d_\sigma)(\|p_\sigma\|/\chi_D - 1)$; and the axis scales follow the separable rank-one plus rank-μ ratio, raised to an exponent modulated by the dimension gate, followed by a geometric-mean renormalization that keeps $\sigma \cdot d$ invariant so the gauge freedom between σ and d does not drift. With all gates at 1 these are precisely the separable CMA-ES updates (Hansen, 2016).

3.4 Fixed distributional machinery

Three mechanisms sit outside the learned part and are shared by every configuration below.

Mirrored sampling. The first $\lceil N/2 \rceil$ noise rows are drawn independently; each remaining row is either the exact antithetic mirror of a drawn row or an independent draw, selected by a gate $\text{clip}(1 - Ns/m, 0, 1)$ where s is the slowest success moving average. Every row is marginally $\mathcal{N}(0, I_D)$ under either branch, and the gate reads no clock: when the population is improving often, mirroring is spent sparingly; when improvements dry up, the draw becomes fully antithetic and the odd component of the weighted noise sum cancels.

Pair-consistent path normalization. Mirroring changes the neutral variance of the weighted noise sum, because a mirrored pair contributes $(w_a - w_b)$ rather than two independent draws. We therefore normalize both evolution paths by

$$\mu_{\text{eff}}^{\text{path}} = \Big(\textstyle\sum_{\text{pairs}} (w_a - w_b)^2 + \sum_{\text{unpaired}} w^2\Big)^{-1}, \tag{3}$$

capped at N, instead of the textbook $\mu_{\text{eff}} = (\sum_i w_i^2)^{-1}$. Using the textbook constant with a mirrored draw biases $\|p_\sigma\|/\chi_D$ below its neutral value, which drives a monotone, landscape-independent step-size decay; Section 4.4 measures exactly that failure.

Capped stall re-expansion. A counter tracks generations without an epoch-best improvement, with patience $120 + 30D/N$ (202.5 at $D = 704$, $N = 256$). On firing, the mean is re-anchored on the monotone incumbent, the paths are reset, and $\sigma \leftarrow \max(\sigma, \min(3\sigma, \sigma_0))$, so the step size can only grow and only up to its initial value. The monotone incumbent is never discarded.

3.5 Compared configurations

Seven configurations were evaluated under the identical fixed evaluator, budget, seeds, and scoring rule. Each row is a single, source-level change against the proposed strategy unless stated otherwise.

- **Proposed.** The anchored residual strategy above, 364 meta-parameters, weights frozen after meta-training.
- **Identity-gate anchor** (decisive control). A single module constant makes the gate function return 1 unconditionally, so all three learned modules become the identity. The remaining algorithm is separable CMA-ES plus the fixed machinery of the previous subsection; the meta-parameters have no effect on any update. This is the paper's central comparison.
- **Isotropic step-size-only baseline.** The axis scales are pinned to 1, deleting the diagonal covariance entirely and leaving a $(\mu/\mu_w, \lambda)$-ES with cumulative step-size adaptation, the learned recombination residual, and the learned scalar gates. Hypothesis: at $D = 704$ with 256 samples the diagonal estimate is badly under-sampled, so learned diagonal covariance transferred from a 2–10-dimensional training distribution might be inert or actively harmful.
- **Unanchored learned strategy.** A self-contained port of a standard learned-evolution-strategy formulation with 142 freshly initialized parameters and no classical anchor, meta-trained and transferred under the same evaluator. It differs from the proposed strategy in parameter count and in architecture simultaneously, so we use it only as motivation for anchoring, not as a controlled measurement of it.
- **No mirrored sampling** (ablation). The mirrored draw is replaced by an unmirrored i.i.d. Gaussian block, with the path normalization consistently reverting to μ_{eff}. Hypothesis: mirroring trades slightly slower early progress for a better-conditioned distribution.

- **No pair-consistent path normalization** (ablation). Mirroring is kept but Equation 3 is replaced by the textbook μ_{eff}. Hypothesis: this mis-specifies the step-size controller and produces a landscape-independent step-size collapse.
- **No stall re-expansion** (ablation). The restart predicate is forced false; the counter and incumbent tracking remain but are inert. Hypothesis: the late-run record-setting comes from restart-driven re-exploration once the step size has been ground down.

3.6 Experimental protocol

Meta-training. The outer optimizer is CMA-ES with population 256, run for 1,500 generations. Each outer generation samples 128 fresh analytic tasks: a function drawn from ten BBOB-style training functions, a dimension drawn uniformly from 2 to 10, a shifted and rotated optimum, and additive Gaussian noise with standard deviation drawn from $[0, 0.1]$. Each candidate runs its inner strategy on every task with population 16 for 50 generations, starting at standard deviation 1.0. A candidate's loss on a task is its best observed clean objective; losses are standardized *within* each task across the candidate population and then aggregated by the *median* over the 128 tasks, so no single task or scale can dominate. Non-finite or invalid tasks receive a large fixed penalty. The outer mean after 1,500 generations is the selected weight vector.

Freeze and transfer. The selected weights are serialized, hashed, and re-loaded under a runtime identity check that binds the source hash, the configuration digest, and the evaluator hash. The strategy is then rebuilt for the 704-dimensional control problem with the deployment population. No further optimization of the weights occurs, and no outer state is carried across.

Reporting. Each configuration is evaluated on the same eight fixed seeds (10,000–10,007). We report the score of Equation 1 together with the standard error over those eight seeds, per-seed win counts against the pinned reference, and paired per-seed differences against the proposed configuration; a paired difference is the honest statistic here because all configurations share seeds, initial conditions, and budget. We also report an exact sign test over the eight seeds, but always alongside the win count, because the two-sided sign-test p-value at $n = 8$ is identical for a configuration that wins $8/8$ and one that loses $0/8$. All configurations completed the full 2,000 generations on the same 8-device hardware allocation within comparable wall-clock times, so no comparison is confounded by truncated runs. Following common practice in metaheuristic experimentation (Osaba et al., 2021), we fix budget, seeds, and the reference implementation across all configurations rather than tuning each separately.

4 Results

4.1 The headline, and what it decomposes into

The proposed strategy scores 1.191 ± 0.019 (mean $\pm$ standard error over eight seeds) and exceeds the pinned same-seed CMA-ES reference on $8/8$ seeds (sign test $p = 0.0078$; paired Cohen's $d = 4.16$). In raw units it improves the mean best-so-far return by 417.5 (95% bootstrap interval [356.7, 486.7], in return units) and reaches the reference's final return in 828 generations, a $2.42\times$ budget saving. Read on its own, this is a successful analytic-to-control transfer of a learned optimizer.

The identity-gate control refuses that reading. Switching the learned modulation off—which, as verified at the source level, makes every gate exactly 1 and reduces the strategy to separable CMA-ES plus the fixed machinery—still scores 1.107 ± 0.028 (Figure 2a). Since the reference is 1.000 by definition, the anchor alone supplies 55.8% of the entire above-reference margin. The paired learning increment over the same eight seeds is $+0.083 \pm 0.016$, positive on $8/8$ seeds (Figure 2b). The correct summary of the run is therefore a decomposition, not a headline: pinned CMA-ES $1.000 \rightarrow$ identity-gate anchor $1.107 \rightarrow$ full method 1.191.

Two further cautions apply to the anchor row itself. Its own advantage over the reference is $7/8$ seeds with sign-test $p = 0.070$, which is not significant at the conventional level; and because configuration selection during development used this same eight-seed protocol, 1.191 should be read as the score of a selected configuration rather than an unbiased estimate of the method.

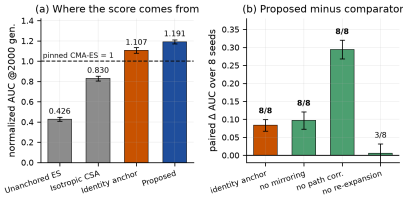


Figure 2: The score decomposes into a large scaffolding term and a small learned term. **(a)** Normalized area under the best-so-far curve at 2,000 generations, with standard errors over eight seeds; the dashed line is the pinned same-seed CMA-ES reference, which is a normalization constant rather than a measured bar. **(b)** Paired per-seed differences of the proposed configuration minus each comparator, annotated with the number of seeds on which the proposed configuration is ahead. Among the four comparators plotted here, removing the pair-consistent path normalization is the largest effect; the isotropic and unanchored configurations of Table 1, which replace the covariance representation and the architecture rather than removing one mechanism, differ by more. Removing the stall re-expansion is a null ($3/8$ seeds).

4.2 The learned increment is budget- and axis-dependent

Figure 3 shows why the decomposition, and not the single number, is the defensible summary.

Budget. The two configurations approach the reference from opposite sides. The proposed configuration starts far above it and decays toward it as the budget grows, from an anytime score of 1.663 at generation 250 to 1.191 at 2,000. The identity-gate anchor starts below the reference at 0.897, crosses it between generations 500 and 1,000, and keeps rising to 1.107 at 2,000, moving further above the reference over exactly the window in which the proposed configuration is falling toward it. The share of the above-reference excess attributable to the learned part therefore falls monotonically, from 86.5% at generation 1,000 to 64.8% at 1,500 and 44.2% at 2,000, while the anchor's share rises correspondingly. Extrapolating the trend is not warranted, but within the measured window the learned contribution is not a fixed property of the strategy—it is a property of the strategy *at a given budget*.

Metric axis. The score of Equation 1 integrates the whole best-so-far curve and therefore rewards fast early progress. On final solution quality the ordering inverts. The proposed configuration finishes at 3488.5 mean best return, the identity-gate anchor at 3709.2: the anchor ends 220.7 higher, and its return curve overtakes the proposed one at generation 1216 and never falls back (Figure 3b). Per-seed final returns were logged only against the pinned reference, which both configurations beat on most seeds ($7/8$ for the anchor, $8/8$ for the proposed configuration), so this reversal rests on the mean gap and the crossover rather than on a paired seed count. The learned modulation buys a faster approach and a worse destination. A study that reported only the anytime score, or only the final return, would have reported opposite conclusions from the same eight runs.

4.3 Meta-learning worked; the transfer is what fell short

It would be easy to conclude that the outer search simply failed to learn anything. It did not. Evaluated on its own training distribution with the same frozen weights, the strategy cuts final log regret from 0.766 to 0.376 on the training functions (-50.8%) and from 1.553 to 1.112 on held-out functions

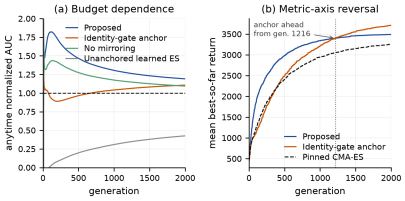


Figure 3: The advantage is conditional on both budget and metric. **(a)** Anytime normalized score as a function of generation for four configurations, with the pinned reference at 1. The two learned anchored configurations start above the reference and decay toward it, while the identity-gate anchor starts below it and rises through it; the unanchored learned strategy improves but never reaches it. **(b)** Mean best-so-far return for the proposed configuration, the identity-gate anchor, and the pinned reference. The anchor is behind for most of the run, passes the proposed configuration at generation 1216, and finishes 220.7 ahead.

(-28.4%), relative to the identity-gate anchor evaluated on exactly the same tasks (Figure 4). For the identity-gate control these two differences are exactly 0.000 by construction, which is the sanity check that the anchor comparison is what it claims to be. The learned weights are also far from their initialization: their Euclidean distance from the anchor point is 20.62, versus 0.0021 for the control whose gates are inert.

The in-distribution gain is nonetheless narrow. Of the eleven instrumented functions, three regress under the frozen weights, and a single training function—the ill-conditioned *attractive sector*—supplies 88.0% of the mean training-set improvement. On a separate 400-generation analytic probe run at deployment scale, the frozen weights are *worse* than the anchor (-1.10 versus -2.24 final log regret). Meta-training thus found a real but concentrated improvement on its own distribution, and that improvement did not broaden into the transfer regime.

4.4 Which components carry the score

Table 1 gives the full picture. Three findings stand out.

The diagonal covariance is essential, not inert. Pinning the axis scales to 1 costs 0.360 paired ($8/8$ seeds) and drops the configuration below the reference to 0.830. The concern that a diagonal estimate from 256 samples in 704 dimensions would be pure noise is not borne out: even a badly under-sampled diagonal, carrying learned and anchored structure, is worth far more than the learned recombination residual and scalar gates that remain without it.

Path normalization is a correctness requirement, not a tuning choice. Removing the pair-consistent normalization of Equation 3 while keeping the mirrored draw costs 0.294 paired ($8/8$ seeds), the largest of the fixed-machinery ablations. Two configurations in Table 1 differ by more—the isotropic baseline at 0.360 and the unanchored strategy at 0.765—but both replace a representation or an architecture wholesale rather than removing a single mechanism, so they are not comparable effect sizes. The mechanism is visible in the trajectory: that configuration's return freezes at 2527.8 from roughly generation 500 onward, exactly the signature of a step-size controller receiving a persistently biased neutral signal. This is best read as a measurement of self-inflicted mis-specification rather than as the contribution of a component—which is precisely why we report the honest mirroring ablation

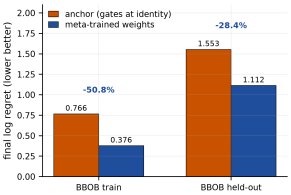


Figure 4: Meta-training succeeded on its own distribution. Final log regret of the frozen weights against the identity-gate anchor on the analytic meta-training functions and on held-out analytic functions (lower is better). The same frozen weights buy only a $+0.083$ paired increment on the 704-dimensional control transfer, and lose on final solution quality, which places the shortfall in transfer rather than in meta-learning.

Table 1: All seven configurations under the identical evaluator, budget, seeds, and scoring rule. Score is the normalized anytime area under the best-so-far curve at 2,000 generations (pinned same-seed CMA-ES $= 1$), with standard error over eight seeds. Δ is the paired per-seed difference of the proposed configuration minus the row, with the number of seeds on which the proposed configuration is ahead. Wins is seeds above the pinned reference. G_{cma} is generations to match the reference's final return; a dash marks configurations that never reached it within 2,000 generations.

Configuration	Score	Δ (paired)	Wins	Final return	G_{cma}
Proposed (anchored residual)	1.191 ± 0.019	—	8/8	3488.5	828
No stall re-expansion	1.185 ± 0.023	0.006 ± 0.026 (3/8)	8/8	3461.1	839
Identity-gate anchor	1.107 ± 0.028	0.083 ± 0.016 (8/8)	7/8	3709.2	1042
No mirrored sampling	1.094 ± 0.022	0.097 ± 0.024 (8/8)	8/8	3286.9	1509
No pair-consistent normalization	0.897 ± 0.021	0.294 ± 0.026 (8/8)	0/8	2527.8	—
Isotropic step-size only	0.830 ± 0.024	0.360 ± 0.019 (8/8)	0/8	2362.5	—
Unanchored learned strategy	0.426 ± 0.020	0.765 ± 0.021 (8/8)	0/8	2007.5	—
Pinned CMA-ES reference	1.000	—	—	3256.9	2000
Initial policy	0.000	—	—	497.4	—

separately. Removing mirroring itself, with the normalization consistently reverted, costs only 0.097 paired—the same order as the $+0.083$ learning increment.

The stall re-expansion is a null. Deleting it changes the score by 0.006 paired, with the proposed configuration ahead on only 3/8 seeds and the paired standard error (0.026) four times the effect; the per-seed differences straddle zero, from -0.136 to $+0.084$. The mechanism's patience is 202.5 generations at $D = 704$, $N = 256$, and on the separate 400-generation analytic probe—run at the same $D = 704$, $N = 256$ scale—the stall counter peaks at 7 and the re-expansion never fires there. We did not instrument firings over the 2,000-generation control runs, and we cannot infer that the mechanism was dormant there: the longest recorded stretch without a strict best-so-far improvement on those runs averages 275 generations and reaches 578, above the patience threshold. The negative result therefore rests on the paired delta alone—whether or not the re-expansion fired on Ant, deleting it does not move the score.

4.5 Anchoring as motivation, not as a controlled measurement

The unanchored learned strategy scores 0.426 and never reaches the reference on any seed, ending at 2007.5 return against the reference's 3256.9. The contrast with 1.191 is the intuition that motivated anchoring in the first place, and Figure 3a shows it is not a slow-start artifact: that configuration's anytime score rises monotonically and still ends far below 1. We deliberately decline to read the 0.765 gap as the causal effect of anchoring. That configuration differs from the proposed one in parameter count (142 versus 364), in module architecture, and in feature design at once; the parameter-matched unanchored comparison that would identify the effect was not run. What the run does establish cleanly is the opposite direction of the same question—that the anchor by itself already accounts for most of what the anchored system achieves.

5 Limitations and Threats to Validity

One meta-training run per configuration. Every configuration was meta-evolved once. The eight seeds are *evaluation* seeds inside that single run, so every standard error we report excludes meta-training-seed variance, which is plausibly the larger source. Reported dispersion is therefore a lower bound on the true uncertainty.

Comparisons confound mechanism with meta-training outcome. Every configuration, including the identity-gate control, was meta-evolved in its own separate run, and three of them—the isotropic step-size-only baseline and the no-mirroring and no-path-normalization ablations—landed at weights measurably different from the proposed configuration's: the recorded distance from the anchor point is 20.62 for the proposed configuration and 19.12, 13.50 and 9.90 for those three. For them, the reported delta mixes "this mechanism was removed" with "a different single meta-training run happened." The no-stall-re-expansion ablation is the exception among the configurations whose gates are active: its recorded distance is 20.62, matching the proposed configuration's to full precision, so for that row the confound does not apply and the delta is closest to a clean mechanism removal. We note this as an observation about the recorded weights and do not use it to argue anything about the other rows. The identity-gate control is immune to the confound by construction, because its meta-parameters provably cannot affect any update—which is another reason it, rather than the ablation table, carries the paper's claim.

One environment, one policy class, one budget regime. All transfer evidence comes from a single Brax Ant task with a tanh-linear policy at $D = 704$. We have shown the conclusion is sensitive to the budget and to the metric; we have not shown it is stable across environments, policy classes, or population sizes. The 2,000-generation window also sits close to where the two curves cross, so the sign of the "learning helps" conclusion is not robust to a modest change in horizon.

Selection and reporting. The eight-seed protocol used to report the headline was also used to select the configuration during development, so the reported score carries a selection bias of unquantified size. The sign-test p-value we quote is direction-agnostic at $n = 8$ and is only interpretable next to the win count. Finally, the analytic probes and the control task are different measurement families, and we have kept them on separate axes rather than merging them into a single figure.

Interpretability of the learned part. Bounded multiplicative residuals around an identity make the learned component unusually legible for a learned optimizer—its worst-case effect is a factor of two on a named quantity—but we have not extracted a human-readable account of *what* the 364 parameters encode, and the gate diagnostics available in this study were recorded only for the configuration in which the gates are inert. Learned components embedded in otherwise transparent algorithms remain black boxes in the sense studied by the explainability literature (Guidotti et al., 2018; Arrieta et al., 2020), and the anchored design mitigates the risk rather than removing it.

6 Conclusion

We asked how much of a frozen-transferred learned evolution strategy's advantage is produced by meta-learning and how much by the classical algorithm it is anchored to, and we built the strategy so that the question has an exact answer. Against a pinned same-seed CMA-ES reference at 1.000, the full method scores 1.191 and the identity-gate anchor—the same code with every learned gate

provably pinned to one—scores 1.107. The anchor supplies 55.8% of the above-reference margin. The remaining increment is real and consistent across seeds, but it shrinks as the budget grows and reverses sign on final solution quality.

The negative result is specific and, we think, more useful than the positive one. Meta-training on cheap analytic functions clearly worked: it halved final log regret in-distribution and improved held-out analytic functions by nearly thirty percent. What did not survive was the transport of that gain to a 704-dimensional control policy, where it degraded into a budget-dependent head start. Anchoring is the design choice that bounds how far that failure can propagate—no gate can move a quantity by more than a factor of two in a generation, and the classical algorithm remains exactly recoverable as a control. It does not guarantee that the anchored system matches that algorithm, and here it did not: the full method finished 220.7 return below the identity-gate anchor. The one unanchored configuration we ran under the same protocol collapsed to 0.426, but it also differs in parameter count and architecture, so we report it as motivation rather than as a measurement of what anchoring contributes—an identification this study does not provide.

The methodological point generalizes beyond evolution strategies, and beyond this benchmark. Whenever a learned component is layered onto a competent classical procedure—and gradient-free optimizers of this family are deployed that way in settings from engineering design to variational quantum algorithms (Cerezo et al., 2021; Hamdani et al., 2019)—the honest experiment is the one that can switch the learned part off exactly, on the same code path, and report the two numbers side by side. When such a control is cheap to build, as it is for zero-initialized bounded residuals, there is little excuse for reporting only the headline.

References

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL http://dx.doi.org/10.1016/j.inffus.2019.12.012.

M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, August 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00348-9. URL http://dx.doi.org/10.1038/s42254-021-00348-9.

Saber Elsayed, Ruhul Sarker, Carlos A. Coello Coello, and Tapabrata Ray. Adaptation of operators and continuous control parameters in differential evolution for constrained optimization. *Soft Computing*, 22(19):6595–6616, 2017. ISSN 1433-7479. doi: 10.1007/s00500-017-2712-6. URL http://dx.doi.org/10.1007/s00500-017-2712-6.

Michael Emmerich, Ofer M. Shir, and Hao Wang. *Evolution Strategies*, pp. 89–119. Springer International Publishing, 2018. ISBN 9783319071244. doi: 10.1007/978-3-319-07124-4_13. URL http://dx.doi.org/10.1007/978-3-319-07124-4_13.

Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Discovering representations for black-box optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, pp. 103–111. ACM, 2020. doi: 10.1145/3377930.3390221. URL http://dx.doi.org/10.1145/3377930.3390221.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional conditional neural processes, 2019. URL https://www.repository.cam.ac.uk/handle/1810/300991.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5): 1–42, August 2018. ISSN 1557-7341. doi: 10.1145/3236009. URL http://dx.doi.org/10.1145/3236009.

Hamid Hamdani, Bouchaïb Radi, and Abdelkhalak El Hami. Optimization of solder joints in embedded mechatronic systems via kriging-assisted cma-es algorithm. *International Journal for Simulation and Multidisciplinary Design Optimization*, 10:A3, 2019. ISSN 1779-6288. doi: 10.1051/smdo/2019002. URL http://dx.doi.org/10.1051/smdo/2019002.

Nikolaus Hansen. The cma evolution strategy: A tutorial, 2016. URL https://arxiv.org/abs/1604.00772.

Essam H. Houssein, Mohamed H. Hassan, Mohamed A. Mahdy, and Salah Kamel. Development and application of equilibrium optimizer for optimal power flow calculation of power system. *Applied Intelligence*, 53(6):7232–7253, 2022. ISSN 1573-7497. doi: 10.1007/s10489-022-03796-7. URL http://dx.doi.org/10.1007/s10489-022-03796-7.

Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application, 2020. URL https://arxiv.org/abs/2009.12836.

Saurav Jha, Dong Gong, Xuesong Wang, Richard E. Turner, and Lina Yao. The neural process family: Survey, applications and perspectives, 2022. URL https://arxiv.org/abs/2209.00517.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996. ISSN 1076-9757. doi: 10.1613/jair.301. URL http://dx.doi.org/10.1613/jair.301.

Robert Tjarko Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, and Sebastian Flennerhag. Discovering attention-based genetic algorithms via meta-black-box optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pp. 929–937. ACM, 2023. doi: 10.1145/3583131.3590496. URL http://dx.doi.org/10.1145/3583131.3590496.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks, 2018. URL https://arxiv.org/abs/1810.00825.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017. URL https://arxiv.org/abs/1707.09835.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference, 2021. URL https://arxiv.org/abs/2112.10510.

Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. A review of population-based metaheuristics for large-scale black-box global optimization—part i. *IEEE Transactions on Evolutionary Computation*, 26(5):802–822, October 2022. ISSN 1941-0026. doi: 10.1109/tevc.2021.3130838. URL http://dx.doi.org/10.1109/tevc.2021.3130838.

Eneko Osaba, Esther Villar-Rodríguez, Javier Del Ser, Antonio J. Nebro, Daniel Molina, Antonio LaTorre, Ponnuthurai Nagaratnam Suganthan, Carlos A. Coello Coello, and Francisco Herrera. A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm and Evolutionary Computation*, 64:100888, 2021. ISSN 2210-6502. doi: 10.1016/j.swevo.2021.100888. URL http://dx.doi.org/10.1016/j.swevo.2021.100888.

Theodore Papamarkou, Jacob Hinkle, M. Todd Young, and David E. Womble. Challenges in markov chain monte carlo for bayesian neural networks. *Statistical Science*, 37(3), August 2022. ISSN 0883-4237. doi: 10.1214/21-sts840. URL http://dx.doi.org/10.1214/21-sts840.

Shahin Rostami and Alex Shenfield. A multi-tier adaptive grid algorithm for the evolutionary multi-objective optimisation of complex problems. *Soft Computing*, 21(17):4963–4979, 2016. ISSN 1433-7479. doi: 10.1007/s00500-016-2227-6. URL http://dx.doi.org/10.1007/s00500-016-2227-6.

Learned CMA-ES Paper (pages 9–12 of 13)

Thomas Philip Rúnarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, 2000. ISSN 1089-778X. doi: 10.1109/4235.873238. URL http://dx.doi.org/10.1109/4235.873238.

Volkmar Sauerland, Claudine von Hallern, Iris Kriest, and Julia Getzlaff. A cma-es algorithm allowing for random parameters in model calibration. *Journal of Advances in Modeling Earth Systems*, 15(8), August 2023. ISSN 1942-2466. doi: 10.1029/2022ms003390. URL http://dx.doi.org/10.1029/2022ms003390.

André Thomaser, Marc-Eric Vogt, Anna V. Kononova, and Thomas Bäck. *Transfer of Multi-objectively Tuned CMA-ES Parameters to a Vehicle Dynamics Problem*, pp. 546–560. Springer Nature Switzerland, 2023. ISBN 9783031272509. doi: 10.1007/978-3-031-27250-9_39. URL http://dx.doi.org/10.1007/978-3-031-27250-9_39.

Joaquin Vanschoren. *Meta-Learning*, pp. 35–61. Springer International Publishing, 2019. ISBN 9783030053185. doi: 10.1007/978-3-030-05318-5_2. URL http://dx.doi.org/10.1007/978-3-030-05318-5_2.

Teppei Yamaguchi, Kento Uchida, and Shinichi Shirakawa. Improvement of sep-cma-es for optimization of high-dimensional functions with low effective dimensionality. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1659–1668. IEEE, December 2022. doi: 10.1109/ssci51031.2022.10022244. URL http://dx.doi.org/10.1109/ssci51031.2022.10022244.

Qiang Yang, Wei–Neng Chen, Jeremiah D. Deng, Yun Li, Tianlong Gu, and Jun Zhang. A level-based learning swarm optimizer for large-scale optimization. *IEEE Transactions on Evolutionary Computation*, 22(4):578–594, August 2018. ISSN 1941-0026. doi: 10.1109/tevc.2017.2743016. URL http://dx.doi.org/10.1109/tevc.2017.2743016.

Zhi-Hui Zhan, Jian-Yu Li, Sam Kwong, and Jun Zhang. Learning-aided evolution for optimization. *IEEE Transactions on Evolutionary Computation*, 27(6):1794–1808, December 2023. ISSN 1941-0026. doi: 10.1109/tevc.2022.3232776. URL http://dx.doi.org/10.1109/tevc.2022.3232776.

Learned CMA-ES Paper (page 13 of 13)

Review


```
{
  "Summary": "The paper presents an anchored residual evolution strategy that augments a separable CMA-ES
  core with three small neural gate modules whose final layers are zero-initialized. This anchoring ensures
  exact recovery of classical CMA-ES during initialization or when gates are forced to identity, enabling a
  clean decomposition of performance gains. The authors meta-evolve 364 parameters on 2-10 D analytic
  functions, freeze them, and transfer to a 704 D Brax Ant policy search. Against a pinned same-seed CMA-ES
  baseline (normalized to 1.0), the full method scores 1.191× while the anchor alone scores 1.107×, showing
  55.8% of the margin stems from the classical backbone. The learned residual adds an early improvement that
  shrinks with budget and even reverses on final return. Detailed ablations isolate the roles of mirroring,
  path normalization, covariance gating, and stall re-expansion, and in-distribution tests confirm genuine
  meta-learning gains on analytic tasks.",
  "Strengths": [
    "Innovative anchored-residual design that exactly recovers CMA-ES at initialization to isolate learned
    contributions.",
    "Rigorous performance decomposition quantifying classical vs. learned gains under the same code path and
    seeds.",
    "Comprehensive ablations (covariance gating, mirroring, path normalization, stall re-expansion) with
    controlled budgets and seeds.",
    "Budget- and metric-dependent analysis showing early learned benefits but late reversal, highlighting
    evaluation pitfalls.",
    "Clear, well-organized writing and reproducible protocol with transparent reporting of negative or
    conditional results."
  ],
  "Weaknesses": [
    "Empirical evaluation is limited to a single high-dimensional control task (Brax Ant) and one policy
    class; generality is untested.",
    "Each configuration was meta-trained only once, so meta-training seed variability and hyperparameter
    sensitivity are not quantified.",
    "The unanchored baseline differs in architecture and parameter count, confounding the isolation of
    anchoring effects.",
    "No intermediate-dimension transfers, comparisons to low-rank/full-covariance ES, or alternative
    learned-optimizer frameworks.",
    "Learned gate modules remain largely a black box with no fine-grained interpretability of what the 364
    parameters encode."
  ],
  "Originality": 3,
  "Quality": 3,
  "Clarity": 4,
  "Significance": 3,
  "Questions": [
    "Can you report performance variance over multiple independent meta-training runs to account for seed
    effects?",
    "Have you evaluated transfer on any other high-dimensional tasks or different policy architectures to
    test generality?",
    "Is it possible to construct a parameter-matched unanchored baseline (same architecture/count) for a
    cleaner anchoring ablation?",
    "Have you tried intermediate-dimension transfers (e.g., 50-100 D) to assess how the learned residual
    scales with problem size?",
    "Can you provide diagnostics on the learned gates (e.g., distributions over generations) to shed light
    on their functional role?"
  ],
  "Limitations": [
    "Scope restricted to one ES variant (separable CMA-ES), one RL environment, and one budget regime;
    limits external validity.",
    "Single-run meta-training underestimates overall uncertainty and may introduce selection bias in
    reported gains.",
    "Ablation setups retrained under new seeds confound mechanism impact with training noise.",
    "Anchoring control relies on multiplicative zero-initialized gates, which may not generalize to all
    learned-optimizer architectures.",
    "Transfer improvement is modest and budget-sensitive, highlighting the need for further architectural or
    domain expansions."
  ],
  "Ethical Concerns": false,
  "Soundness": 3,
  "Presentation": 4,
  "Contribution": 3,
  "Overall": 6,
  "Confidence": 4,
  "Decision": "Accept"
}
```

C.3. Error Diffusion ALIFE Paper

Diffusing Blame: Task-Dependent Credit Assignment in Biologically Plausible Dual-Stream Networks

Yutaro Yamada, Luca Grillotti, Rujikorn Charakorn, Sebastian Risi, David Ha, Robert Tjarko Lange

Sakana AI, Tokyo, Japan
yutaro.yamada.y@gmail.com, robert@sakana.ai

Abstract

Biological neural circuits obey Dale's principle: each neuron's synapses are uniformly excitatory or inhibitory. Artificial networks that respect this constraint must coordinate separate excitatory and inhibitory populations, fundamentally changing how credit is assigned during learning. Several biologically plausible learning rules avoid backpropagation's weight transport requirement, but it has been difficult to achieve strong performance under Dale's principle beyond MNIST. Error Diffusion (ED) was originally proposed in a dual-stream excitatory/inhibitory architecture, where learning is driven by routing global error signals to all layers without transporting transposed forward weights or relying on random feedback matrices. Whether such a rule can scale under Dale's principle across both supervised classification and reinforcement learning remains unknown. Here, we introduce modulo error routing to extend Error Diffusion beyond binary classification, and show that a dual-stream excitatory/inhibitory architecture trained with this method achieves 96.7% on MNIST and establishes a 61.7% baseline on CIFAR-10, demonstrating that representation learning is possible even when strictly enforcing Dale's principle. For the classification setting, we introduce three domain-specific innovations: layer-specific sigmoid widths, batch-centered class error signals, and asymmetric initialization, and ablation analysis reveals that their relative importance *reverses* between MNIST and CIFAR-10, exposing task-dependent credit-assignment bottlenecks invisible to single-benchmark evaluation. In reinforcement learning, we integrate ED with Proximal Policy Optimization (PPO) and evaluate it on continuous-control tasks in Google Brax and on Craftax, an open-ended exploration task. We show that ED-PPO achieves competitive performance relative to Direct Feedback Alignment, a backpropagation-free baseline.

Introduction

In biological neural circuits, the separation of excitatory and inhibitory neuronal populations is a fundamental organizational principle, formalized as Dale's principle: each neuron releases the same neurotransmitter at all of its synapses (Dale, 1935; Eccles, 1976; Strata and Harvey, 1999). This constraint shapes cortical computation through balanced excitation and inhibition (Markram et al., 2004). Artificial neural networks, by contrast, freely assign positive and negative weights to any connection, a simplification that enables efficient credit assignment via backpropagation but bears little resemblance to biological learning. Training deep networks with backpropagation requires the backward pass to use exact transposes of forward weight matrices, the *weight transport problem*, which lacks biological support (Crick, 1989; Bengio et al., 2015). Several alternatives address this concern. Feedback Alignment (FA) replaces backward weights with fixed random matrices (Lillicrap et al., 2016), Direct Feedback Alignment (DFA) routes output errors directly to each hidden layer through fixed random matrices (Nøkland, 2016), and predictive coding frameworks approximate backpropagation through local Hebbian updates (Whittington and Bogacz, 2017; Sacramento et al., 2018). While these methods avoid weight transport, none enforce Dale's principle: all allow arbitrary sign weights. Error Diffusion (ED), originally proposed by (Kaneko, 2000), is a local learning rule in which weight updates depend on presynaptic activity, a postsynaptic activation derivative, and a single global error sign (Fujita, 2026). This locality makes ED naturally compatible with biological constraints, but prior work demonstrated its effectiveness only on binary classification and a simple classification task like MNIST (Fujita, 2026). The central question we address is whether a biologically plausible architecture that enforces Dale's principle can achieve competitive performance across both supervised classification and reinforcement learning, and what the task-dependent importance of its components reveals about credit assignment under biological constraints. We make three contributions:

1. A dual-stream excitatory/inhibitory ED architecture that maintains non-negative weights across classification (MNIST, CIFAR-10) and reinforcement learning (Brax locomotion tasks, Craftax) by extending the original ED via modulo error routing.
2. For the classification setting, three domain-specific innovations: layer-specific sigmoid widths, batch-centered class error, and asymmetric initialization, whose cross-task ablation reveals that credit-assignment bottlenecks shift qualitatively with task difficulty.
3. Cross-domain evaluation shows that ED-PPO achieves returns comparable to backpropagation-based and DFA-based PPO on continuous-control tasks such as Ant, Humanoid, and HalfCheetah. On the more complex Craftax task, ED-PPO achieves performance comparable to DFA.

©2026 Yutaro Yamada, Luca Grillotti, Rujikorn Charakorn, Sebastian Risi, David Ha, Robert Tjarko Lange. Published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Related Work

Biologically plausible learning rules. The weight transport problem has motivated a family of alternatives to backpropagation (Crick, 1989; Bengio et al., 2015). Feedback Alignment uses fixed random backward weights (Lillicrap et al., 2016), and Liao et al. (2016) showed that approximate weight symmetry can support learning in deeper networks. DFA projects output errors directly to hidden layers (Nøkland, 2016), and Launay et al. (2020) demonstrated that DFA scales to modern architectures, including transformers and convnets. Difference target propagation computes layer-wise learning targets instead of gradients (Lee et al., 2015). Predictive coding provides a complementary framework through local Hebbian updates (Whittington and Bogacz, 2017), and dendritic cortical microcircuit models approximate backpropagation with segregated dendrites (Sacramento et al., 2018). Bartunov et al. (2018) found that biologically motivated algorithms generally struggle to scale beyond simple benchmarks, and Richards et al. (2019) argued that biological plausibility and computational performance need not be in tension if the right architectural principles are identified. Filipovich et al. (2022) demonstrated DFA on photonic hardware, showing that bioplausible rules can enable novel computing substrates.

Dale's principle in neural networks. Song et al. (2016) introduced a framework for training excitatory-inhibitory recurrent networks with Dale's principle, separating synaptic magnitudes from signs by parameterizing weights as $W = W^+ \odot M$, where $W^+ \geq 0$ contains trainable connection magnitudes and M is a fixed sign mask encoding each unit's excitatory or inhibitory identity. Cornford et al. (2021) proposed Dale's ANNs, feedforward architectures with separate excitatory and inhibitory populations that can learn comparably to standard ANNs by treating inhibition as a normalization mechanism. Grossberg (1987) explored competitive learning in networks with separate excitatory and inhibitory interactions. Error Diffusion (Fujita, 2026; Kaneko, 2000) is also naturally Dalean, representing positive and negative components through separate non-negative populations while avoiding both weight transport and random feedback. Building on this framework, our work extends Error Diffusion to multi-class classification and reinforcement learning.

Reinforcement learning without backpropagation. Evolution Strategies (ES) provide a gradient-free alternative to policy gradient methods (Salimans et al., 2017), but scale poorly with parameter count. Proximal Policy Optimization (PPO) is a widely used policy gradient method (Schulman et al., 2017) that typically relies on backpropagation for gradient computation. Biologically plausible alternatives for RL credit assignment have been proposed—reward-modulated Hebbian rules can solve decision-making tasks by gating synaptic updates with a global reward signal (Pfeiffer et al., 2010), and Izhikevich (2007) showed that linking spike-timing-dependent plasticity with dopamine signaling addresses the distal reward problem—but these methods have not been demonstrated with deep networks on continuous-control benchmarks. We replace backpropagation in PPO's gradient computation with Error Diffusion, evaluating on Brax physics tasks (Freeman et al., 2021) and Craftax (Matthews et al., 2024).

Method

Dual-Stream Architecture

We adopt the original Error Diffusion formulation proposed by (Kaneko, 2000). To enforce Dale's principle, we split each layer into an excitatory ($\mathbf{p}$) and inhibitory ($\mathbf{n}$) stream. The forward pass for layer i computes:

$$\mathbf{p}_i = \phi_i(+\mathbf{p}_{i-1}W_{pp} - \mathbf{n}_{i-1}W_{np} + \mathbf{b}_p)$$
$$\mathbf{n}_i = \phi_i(+\mathbf{n}_{i-1}W_{nn} - \mathbf{p}_{i-1}W_{pn} + \mathbf{b}_n)$$

where all weight matrices $W_{pp}, W_{np}, W_{nn}, W_{pn} \geq 0$ element-wise, bias parameters $\mathbf{b}_p$ and $\mathbf{b}_n$ are not necessary non-negative, ϕ_i is a layer-specific activation function. The negation signs in front of W_{np} and W_{pn} are structural (hardcoded), ensuring that cross-stream connections are inhibitory while all learnable parameters remain non-negative. This dual-stream design requires four weight sub-matrices per layer, resulting in $\sim$4$\times$ more parameters than an unconstrained single-stream network of the same width (e.g., $\sim$32M vs $\sim$8M for DFA on the same architecture).

Error Diffusion Update Rule

The ED update replaces backpropagated layerwise errors with an output-space error signal that is routed directly to hidden units. While the original formulation was developed for binary classification (Kaneko, 2000; Fujita, 2026), we extend ED to multi-output prediction by assigning each hidden unit a fixed output channel. For a hidden unit i, we define $r(i) = i \bmod C$, where C is the output dimension, and use the routed error component $s_{r(i)}$ as its learning signal.

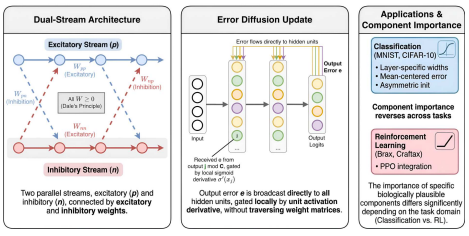


Figure 1: Overview of the dual-stream Error Diffusion framework. **Left:** The excitatory/inhibitory architecture maintains separate streams ($\mathbf{p}$, $\mathbf{n}$) with four non-negative weight matrices per layer, enforcing Dale's principle structurally. **Center:** The Error Diffusion update broadcasts output error directly to hidden layers without weight transport or random feedback matrices. **Right:** The shared architecture is applied to classification (with domain-specific sigmoid widths, batch-centered class error, and asymmetric initialization) and reinforcement learning (with PPO integration).

(For other design choices of this error routing and why, see Figure 4)

For a mini-batch, let $S \in \mathbb{R}^{B\times C}$ denote the output error signal, and let $M \in \{0,1\}^{H\times C}$ be the fixed routing matrix with $M_{ic} = 1$ iff $r(i) = c$. The routed hidden error is

$$R = SM^\top, \tag{1}$$

and the corresponding matrix of local postsynaptic drives is

$$U_p = \phi'(Z_p) \odot R, \tag{2}$$

where $Z_p \in \mathbb{R}^{B\times H}$ is the positive-stream preactivation matrix.

Let $A_p \in \mathbb{R}^{B\times K}$ denote the positive-stream presynaptic activations feeding the layer. In fully connected notation, the positive-to-positive ED update is

$$\Delta W_{pp} \propto A_p^\top U_p. \tag{3}$$

In the supervised image classifiers, $C = 10$ and S is the batch-centered one-vs-all classification error. In Craftax policy networks, C is the number of discrete action logits. In MuJoCo policy networks, C is the policy distribution parameter dimension for the TanhNormal policy.

Each local weight update is proportional to presynaptic activity, and this routed postsynaptic drive, with stream-specific signs for the dual positive/negative pathways. This deterministic modulo routing provides coarse output-associated credit assignment without transporting forward weights or using random feedback matrices. Unlike DFA, whose fixed feedback matrices are random, ED uses a structured correspondence between hidden units and output dimensions.

Classification-Specific Innovations

For the classification setting (MNIST, CIFAR-10), we introduce three domain-specific innovations that address failure modes specific to multi-class classification under Dale's principle. These three innovations are specific to the classification setting and are not used in the RL extension.

Layer-specific sigmoid widths. In the classification architecture, $\phi_i(z) = 1/(1 + e^{-2z/\alpha_i})$ with layer-type-specific sigmoid width α_i. The vanishing gradient problem in sigmoid networks is well documented (Roodschild et al., 2020; Ding et al., 2018), and various parameterized activations have been proposed (Szandała, 2020; Hammad, 2024). Since the sigmoid derivative directly gates the error signal in ED (Eq. 3), gradient attenuation is especially severe: post-hoc analysis reveals a $25\times$ decay from output to first hidden layer. Wider sigmoids (larger α_i) maintain larger derivatives, preventing premature saturation. We set $\alpha = 3.0$ for CIFAR-10 convolutional layers and $\alpha = 6.0$ for fully connected layers, including the output layer; MNIST uses only the $\alpha = 6.0$ fully connected setting.

Batch-centered class error signals. For the 10-way classification tasks, the ED update uses independent sigmoid output activations rather than a softmax. For a mini-batch of size B, we form a one-vs-all output error $E_{b,c} = \mathbf{1}[y_b = c] - \hat{y}_{b,c}$, where $\hat{y}_{b,c}$ is the positive-stream output activation for class c. Since each example contributes one target class and nine non-target classes, the raw one-vs-all error can contain a strong class-wise offset, especially early in training. We therefore subtract the mini-batch mean separately for each class: $\tilde{E}_{b,c} = E_{b,c} - \frac{1}{B}\sum_{b'=1}^{B} E_{b',c}$. This makes the error signal zero-mean across the batch for every class, reducing persistent suppression or excitation of class channels caused by the one-vs-all target imbalance. The centered signal is then applied directly to the output layer, with the negative stream receiving the opposite sign; hidden fully connected units and convolutional channels receive class-routed copies of the same centered error.

Asymmetric E/I initialization. For convolutional and fully connected layers, weight parameters are initialized from a non-negative uniform distribution in $[0, 1)$ and scaled by the inverse square root of the fan-in. Hidden excitatory weights (W_{pp}, W_{nn}) are then scaled by $1.5\times$, while inhibitory weights (W_{np}, W_{pn}) are scaled by $0.5\times$, giving an expected E/I scale ratio of $3:1$. The final fully connected output layer uses symmetric initialization ($1.0\times$ for both excitatory and inhibitory weights) to avoid output saturation.

RL Extension: ED-PPO

For reinforcement learning, we integrate a Dale-constrained dual-stream Error Diffusion (ED) architecture into PPO. The policy and value networks are both dual-stream MLPs with separate positive and negative pathways and non-negative synaptic weights. Each layer computes excitatory-minus-inhibitory preactivations for the positive and negative streams, applies a width-scaled sigmoid nonlinearity, and combines the final streams as $\hat{y} = y^+ - y^-$. Both Dale implementations initialize the two streams symmetrically from the same observation, $\mathbf{p}_0 = \mathbf{n}_0 = x$, after optional stream normalization, rather than splitting inputs with ReLU.

ED replaces backpropagation through hidden layers after the PPO's objective function supplies an output-level error signal. For vector-valued policy outputs, the error is routed to hidden units by output class/channel assignment; for the scalar value network, the error is broadcast to all hidden units. All weight matrices are initialized non-negative using absolute Gaussian samples, with separate scaling for excitatory and inhibitory pathways, and weights are clamped non-negative after optimizer updates.

Figure 1 summarizes the error diffusion training procedure and methodological contributions.

Experimental Protocol

Classification. We evaluate six variants on MNIST and CIFAR-10: (1) the **proposed ED** with all three innovations, (2) a **DFA baseline** using fixed random feedback matrices without Dale's constraints, (3) the **seed ED** (the dual-stream architecture without any of the three innovations, i.e., using uniform sigmoid width $\alpha = 1.0$, raw error signals, and symmetric initialization), and three ablations removing one innovation each: (4) **no batch-centered class error**, (5) **symmetric init**, and (6) **uniform sigmoid width**. Each variant is trained for 250 epochs with five random seeds per task (60 runs total). For classification, weights are clamped to a floor of 10^{-4} after each update to enforce Dale's principle.

Reinforcement learning. We compare ED-PPO against BP-PPO (standard backpropagation), DFA-PPO (Direct Feedback Alignment) (Nøkland, 2016), and ES (Salimans et al., 2017) on three Brax (Freeman et al., 2021) locomotion tasks (Ant, HalfCheetah, Humanoid). On Craftax (Matthews et al., 2024), we compare against BP-PPO and DFA-PPO. Additionally, we evaluate a non-Dalean variant of ED-PPO, in which weights are not constrained to be non-negative, and denote it as "ED-PPO (non-Dalean)". Each algorithm is evaluated with five random seeds per environment. We report the final episode reward (the reward at the last evaluation checkpoint) averaged over seeds; statistical comparisons use Welch's t-test with $\alpha = 0.05$. The RL experiments use a separate training and evaluation pipeline from the classification experiments, with both pipelines sharing the same dual-stream ED architecture but differing in activation functions, normalization, and optimization procedure as described above.

Results

Classification: Main Comparison

Figure 2 summarizes accuracy across all six classification variants. The proposed ED achieves $96.7 \pm 0.1\%$ on MNIST and $61.7 \pm 0.7\%$ on CIFAR-10, a substantial improvement over seed ED ($50.4 \pm 9.8\%$ and $11.6 \pm 2.2\%$). DFA achieves higher accuracy on both tasks (97.6% and 69.1%) but violates Dale's principle, requiring $\sim$2.84M negative weights. The accuracy gap between ED and DFA widens from 0.9 pp on MNIST to 7.4 pp on CIFAR-10, suggesting that the cost of maintaining non-negative weights grows with task difficulty. We note that $\sim$62% test accuracy on CIFAR-10 is still far from competitive compared to traditional gradient-based methods. That being said, this is the first time ED has been successfully applied to convolutional neural networks. Previously, Fujita (2026) obtained $\sim$55.2% test accuracy on CIFAR-10 using a purely feedforward MLP with flattening.

Classification: Cross-Task Ablation Reversal

In the classification setting, ablation analysis reveals that the importance hierarchy of the three innovations reverses be-

Table 1: Ablation effect sizes (Δ = proposed ED minus ablation, in percentage). Negative values indicate accuracy loss.

Removed Component	MNIST Δ	CIFAR-10 Δ
Batch-Centered Error	-0.3	-47.9
Layer-Specific Widths	-71.4	-15.1
Asymmetric Init	$+0.0$	-5.5

tween tasks (Figure 2, Table 1).

MNIST. Removing layer-specific sigmoid widths is catastrophic (96.7% $\rightarrow$ 25.3%, $\Delta = -71.4$ pp), collapsing accuracy to near-chance. Removing batch-centered class error causes only $\Delta = -0.3$ pp, and symmetric initialization has no measurable effect ($\Delta = +0.0$ pp). Gradient flow regulation is the sole bottleneck on this easier task.

CIFAR-10. The hierarchy reverses. Removing batch-centered error is now the most destructive ablation (61.7% $\rightarrow$ 13.8%, $\Delta = -47.9$ pp), causing four of five seeds to collapse. Uniform sigmoid width causes $\Delta = -15.1$ pp, and symmetric initialization causes $\Delta = -5.5$ pp. All three innovations contribute, but their relative ordering changes fundamentally.

Interpretation. This reversal reflects qualitatively different credit-assignment bottlenecks. MNIST's well-separated features allow learning even without error centering; but without wide sigmoids, the derivatives vanish entirely. On CIFAR-10, higher inter-class similarity makes the 9:1 error imbalance overwhelming: batch-centering becomes essential to prevent uniform output suppression. The reversal demonstrates that evaluating biologically plausible methods on a single benchmark may obscure critical trade-offs.

RL: Cross-Domain Generalization

Figures 3 and 5 report performance across four RL tasks. ED-PPO is strongest on HalfCheetah, where it outperforms BP-PPO (5494 $\pm$ 691 vs 3520 $\pm$ 485; $p < 0.001$), ED-PPO (non-Dalean) ($p = 0.028$), and ES ($p = 0.003$), while matching DFA-PPO (5581 $\pm$ 359). On Ant it is on par with both PPO variants (6891$\pm$835), significantly exceeding only ES ($p = 0.004$). On Humanoid (6670 $\pm$ 2592) and Craftax (20.9 $\pm$ 2.9) it trails BP-PPO and ED-PPO (non-Dalean), though the Craftax gap to the latter is not significant.

ED-PPO (non-Dalean) also performs competitively across the same four environments, on par with BP-PPO on Humanoid (9804 $\pm$ 1144 vs 8478 $\pm$ 3252). On Ant, ED-PPO (non-Dalean) achieves a higher mean (7616 $\pm$ 2031 vs 6740$\pm$1781) but the difference is not significant due to high variance. On HalfCheetah, ED-PPO (non-Dalean) achieves a similar score (3498 $\pm$ 1703 vs 3520$\pm$ 485). Notably, DFA-PPO on Craftax achieves only 19.8$\pm$1.5, significantly below ED-PPO (non-Dalean) ($p = 0.010$), demonstrating that random feedback pathways that suffice for simpler tasks fail on complex, open-ended environments.

ED-PPO (non-Dalean) exhibits higher variance than BP-PPO on Ant and HalfCheetah, though on Humanoid its variance is lower (1144 vs 3252), suggesting that the ED gradient signal can introduce additional stochasticity. The Craftax shortfall (-4.0, $p = 0.002$) and higher variance across environments suggest that the coarse modular error routing of ED may be less reliable for tasks requiring fine-grained temporal credit assignment or stable convergence.

DFA Across Domains

DFA provides an informative comparison point across both classification and RL. In classification, DFA achieves the highest accuracy (97.6% MNIST, 69.1% CIFAR-10) but violates Dale's principle. On Craftax, DFA-PPO is the weakest method (19.8 vs BP-PPO 27.0, ED-PPO 23.0), demonstrating that random feedback pathways that work for supervised learning can fail for complex RL. This cross-domain pattern—DFA strong on classification, weak on complex RL—parallels the finding that method components assume different importance under different task demands.

Post-hoc Analysis

Surrogate gradient attenuation. Post-hoc analysis of training dynamics on CIFAR-10 (Figure 6a) reveals local surrogate gradient magnitudes dropping from 6.6×10^{-3} at the output layer to 2.7×10^{-4} at the first hidden layer—a $25\times$ attenuation. This pattern is visible from epoch 3, indicating a structural property of sigmoid-gated error diffusion that motivates layer-specific widths. Per-layer learning rate multipliers ($3.0\times$ early, $0.5\times$ output) partially compensate but cannot substitute for wider sigmoid derivatives. The near-zero generalization gap (0.98%) despite ~32M nominal parameters is consistent with underfitting rather than overfitting, but the cause is ambiguous: the 37.3% of weights at floor (10^{-4}) reduces effective capacity well below the nominal parameter count, so the limitation may reflect constrained capacity rather than (or in addition to) credit assignment quality.

Emergent E/I balance. Weight-level E/I ratios (Figure 6b) reveal that training drives the 3:1 asymmetric initialization toward near-balanced ratios ($\sim$1.0) in hidden layers. A depth-dependent gradient emerges: the first layer reaches near-perfect balance (E/I = 1.03), the second becomes slightly inhibitory-dominant (0.90), and the third develops the strongest inhibitory bias (0.81). This increasing inhibition with depth is loosely consistent with biological observations that inhibitory circuitry and E/I balance vary systematically across cortical hierarchies (Markram et al., 2004), though weight-level E/I ratios are an indirect proxy for biological E/I balance, which involves cell counts, firing rates, and synaptic strengths jointly. The convergence toward balance explains the task-dependent importance of asymmetric initialization: on MNIST, balance is reached quickly regardless of initial conditions ($\Delta = +0.0$ pp); on CIFAR-10, the asymmetric head start prevents early instability ($\Delta = -5.5$ pp).

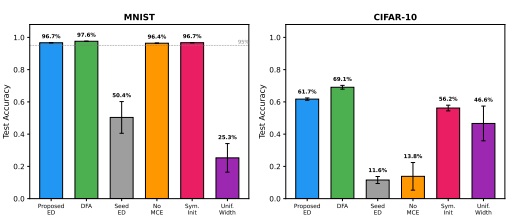
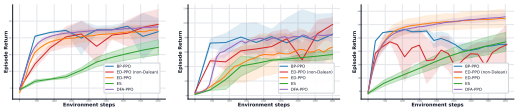


Figure 2: Classification accuracy across all six variants on MNIST (left) and CIFAR-10 (right). Error bars: ± 1 std over 5 seeds. The ablation importance hierarchy reverses between tasks: layer-specific widths dominate on MNIST, batch-centered class error dominates on CIFAR-10.



(a) Ant (b) Humanoid (c) HalfCheetah

Figure 3: Episode return across three RL environments. Error bars indicate ± 1 standard deviation over 5 seeds.

Implicit sparsity. The non-negative weight floor induces substantial implicit sparsity: 37.3% of weights reach the floor (10^{-4}) after training. Inhibitory (cross-stream) FC connections are pruned most aggressively (up to 68.8% at the floor), compared to 26–49% for excitatory connections. Convolutional layers are much less affected (<1% to 18%). This asymmetric pruning suggests that the non-negative Dale-style parameterization introduces a structured capacity bottleneck that preferentially suppresses inhibitory connections in deeper layers.

Discussion

Summary. The central finding of this work is that the dual-stream ED architecture is viable across both classification and reinforcement learning, but the auxiliary mechanisms required for competitive performance are domain-dependent. In classification, batch-centered class error and layer-specific sigmoid widths are critical stabilizers whose relative importance reverses between MNIST and CIFAR-10. In RL, the same dual-stream architecture with ReLU activations and RMS normalization, without sigmoid widths or batch-centering, achieves competitive performance on locomotion tasks, showing comparable but variable rewards on Ant, Humanoid, and HalfCheetah.

Implications for biological plausibility research. Single-benchmark evaluations can produce misleading conclusions about which architectural features matter. The ablation reversal between MNIST and CIFAR-10 demonstrates that a component critical for one task can be negligible for another, and the DFA cross-domain pattern (strong on classification, weak on complex RL) reinforces this point. The emergent E/I balance convergence, from 3:1 toward $\sim$1:1 with a depth-dependent inhibitory gradient, provides a connection to biological self-organization. Cortical circuits are known to develop balanced E/I ratios during maturation through intrinsic and synaptic homeostatic mechanisms (Markram et al., 2004; Turrigiano, 2011), and our results suggest that a similar homeostatic process can emerge from gradient-driven learning under Dale's principle constraints, without explicit balance-enforcing mechanisms. The success of ED-PPO on locomotion tasks despite using ReLU rather than sigmoid activations raises an interesting question about the relationship between activation function choice and credit assignment quality. In classification, the sigmoid derivative explicitly gates the error signal, making the width parameter critical. In RL, ReLU's unbounded positive derivative may provide a natural gradient pathway that avoids the attenuation problem without requiring layer-specific tuning. This activation-dependent difference in error flow may partially explain why the classification-specific innovations are unnecessary in the RL setting.

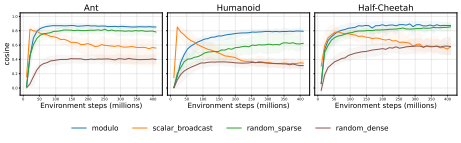


Figure 4: Cosine similarity between the ED local update direction and the BP gradient direction, restricted to policy trunk parameters and excluding the policy in a dual-stream architecture, over PPO training on Ant, Humanoid, and Half-Cheetah. Lines show the mean over 5 seeds; shaded regions show the seed-wise min-max range. `modulo` uses deterministic one-output-per-hidden-unit routing, `random_sparse` uses random one-output-per-hidden-unit routing, `scalar_broadcast` sends the same normalized sum of output errors to every hidden unit, and `random_dense` denotes the normalized dense random feedback variant used in (Launay et al., 2019). Higher cosine indicates closer agreement between ED and BP trunk parameter-gradient directions.

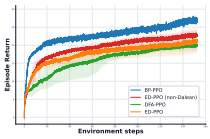
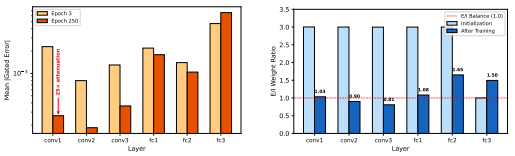


Figure 5: Episode return on Craftax comparing ED-PPO, BP-PPO, and DFA. Error bars: ± 1 std over 5 seeds.

Limitations. The classification innovations (sigmoid widths, MCE, asymmetric init) are specific to the classification setting; we do not claim they transfer to RL. ED-PPO shows substantially higher variance than BP-PPO across all environments, suggesting that the coarse error routing introduces stochasticity that may limit reliability. The Craftax shortfall (-4.0 reward vs BP-PPO) indicates that ED's modular credit assignment may be insufficient for tasks requiring fine-grained temporal reasoning. The accuracy gap between ED and DFA on classification (-0.9 to -7.4 pp) quantifies the current cost of enforcing Dale's principle. On Craftax, DFA underperforms ED-PPO, suggesting that the cost of Dale's principle is task-dependent. Finally, the dual-stream architecture's $\sim$4$\times$ parameter overhead may itself limit effective capacity, particularly on classification where 37.3% of weights are pruned to the floor. From the perspective of adaptive computation, the emergent behaviors we observe—E/I balance self-organization, asymmetric pathway pruning, and task-dependent component criticality—suggest that biologically constrained architectures may develop internal regulatory mechanisms analogous to those found in living systems.

Future Work. Several directions follow naturally from the results presented here. First, Dale's principle may offer hardware advantages by constraining synaptic weights



(a) Per-layer local surrogate-gradient magnitudes at epochs 3 and 250 on CIFAR-10 (log scale). The 25$\times$ attenuation from output to first hidden layer motivates layer-specific sigmoid widths.

(b) E/I weight ratio at init (3:1 for hidden layers) vs. after 250 epochs. Hidden layers converge toward balance; deeper layers become inhibitory-dominant.

Figure 6: Post-hoc analyses of the proposed ED model on CIFAR-10. (a) Local surrogate gradient attenuation across layers. (b) Emergent E/I balance from asymmetric initialization toward biological-like equilibrium.

themselves to be non-negative. In our Error Diffusion formulation, negative contributions are still required, but their sign is determined by fixed excitatory/inhibitory pathway structure rather than by arbitrary signed weights. Thus, the benefit is not the elimination of subtraction or differential circuitry altogether, but the replacement of unconstrained signed synapses with non-negative synaptic magnitudes and fixed-sign routing. This may be especially relevant for analog, photonic, or synapse-device-based neuromorphic substrates, where physical synaptic elements often naturally encode non-negative quantities, while sign can be implemented at the level of population identity, optical/electrical phase, or excitatory/inhibitory summation. This complements prior demonstrations of bioplausible rules on photonic hardware (Filipovich et al., 2022). Second, the implicit sparsity we observe suggests that Dale-compliant training may yield model compression "for free," since weights bounded below at zero cannot recover once suppressed, turning the floor into a natural pruning mechanism that could be exploited with structured sparsity kernels at inference time. Third, we conjecture that the dual-stream architecture may be unusually well-suited to continual and open-ended learning settings, where catastrophic forgetting is a central obstacle: the dedicated inhibitory stream provides a structural mechanism for dampening large gradient excursions, and the sign constraint prevents the kind of unconstrained weight sign flips that are thought to contribute to representational overwriting. Evaluating ED on sequential task streams and lifelong RL benchmarks is a natural next step. Fourth, the segregation of computation into excitatory "amplifiers" and inhibitory "suppressors" offers a potential interpretability handle that standard networks lack: when a model makes an error, one can in principle trace whether the mistake originated from a feature being falsely amplified by the excitatory stream or insufficiently suppressed by the inhibitory stream, providing a more mechanistic form of attribution than gradient-based saliency on sign-unconstrained networks. Finally, closing the accuracy gap to DFA and BP-PPO on the hardest tasks, whether through richer error-routing schemes, learned (rather than fixed modular) output-to-hidden projections, or hybrid activation functions that preserve sigmoid locality while mitigating attenuation, remains the most direct path toward making Dale-compliant learning competitive with unconstrained backpropagation.

Conclusion

We demonstrated that a dual-stream excitatory/inhibitory architecture trained with Error Diffusion achieves competitive performance across both supervised classification and reinforcement learning while maintaining non-negative weights consistent with Dale's principle. In the classification setting, three domain-specific innovations enable scaling from binary to 10-class problems, and their cross-task ablation reversal reveals that credit-assignment bottlenecks shift qualitatively with task difficulty. In reinforcement learning, the same core architecture integrated with PPO achieves comparable rewards to backpropagation-based PPO on locomotion tasks, without requiring the classification-specific stabilizers, though with substantially higher variance. The cross-domain pattern, competitive performance with domain-dependent auxiliary mechanisms, suggests that biologically plausible learning rules need not be monolithic: a shared architectural core can support diverse task demands when augmented with appropriate domain-specific components. The accuracy gap relative to BP-PPO on the hardest tasks quantifies the current cost of Dale's principle compliance and provides concrete benchmarks for future work on narrowing this gap while preserving biological fidelity.

Acknowledgement

Generative AI tools were used in the preparation of the manuscript, including text editing, code generation, image generation, and data analysis assistance. Additionally, we leveraged AI to generate ideas, run autonomous experimentation, and refine hypotheses. All claims, code implementation, manuscript writing, and figures were either reviewed by or created by the authors.

References

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Cornford, J., Kalajdzievski, D., Leite, M., Lamarquette, A., Kullmann, D. M., and Richards, B. A. (2021). Learning to live with dale's principle: {ANN}s with separate excitatory and inhibitory units. In *International Conference on Learning Representations*.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Dale, H. (1935). Pharmacology and nerve-endings. *Proceedings of the Royal Society of Medicine*.

Ding, B., Qian, H., and Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841.

Eccles, J. C. (1976). From electrical to chemical transmission in the central nervous system: the closing address of the sir henry dale centennial symposium cambridge, 19 september 1975. *Notes and records of the Royal Society of London*, 30(2):219–230.

Filipovich, M. J., Guo, Z., Al-Qadasi, M., Marquez, B. A., Morison, H. D., Sorger, V. J., Prucnal, P. R., Shekhar, S., and Shastri, B. J. (2022). Silicon photonic architecture for training deep neural networks with direct feedback alignment. *Optica*, 9(12):1323–1332.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*.

Fujita, K. (2026). A diagnostic evaluation of neural networks trained with the error diffusion learning algorithm. *Discover Artificial Intelligence*.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63.

Hammad, M. (2024). Deep learning activation functions: Fixed-shape, parametric, adaptive, stochastic, miscellaneous, non-standard, ensemble. *arXiv preprint arXiv:2407.11090*.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452.

Kaneko, I. (2000). Sample programs of error diffusion learning algorithm. https://web.archive.org/web/20000306212433/http://village.infoweb.ne.jp/~fwhz9346/ed.htm. [Online; accessed 13-April-2026].

Launay, J., Poli, I., Boniface, F., and Krzakala, F. (2020). Direct feedback alignment scales to modern deep learning tasks and architectures. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Launay, J., Poli, I., and Krzakala, F. (2019). Principled training of neural networks with direct feedback alignment.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. In Appice, A., Rodrigues, P. P., Santos Costa, V., Soares, C., Gama, J., and Jorge, A., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 498–515, Cham. Springer International Publishing.

Liao, Q., Leibo, J., and Poggio, T. (2016). How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276.

Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., and Wu, C. (2004). Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience*, 5(10):793–807.

Matthews, M., Beukman, M., Ellis, B., Samvelyan, M., Jackson, M., Coward, S., and Foerster, J. (2024). Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29.

Pfeiffer, M., Nessler, B., Douglas, R. J., and Maass, W. (2010). Reward-modulated hebbian learning of decision making. *Neural computation*, 22(6):1399–1444.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., et al. (2019). A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770.

Roodschild, M., Gotay Sardiñas, J., and Will, A. (2020). A new approach for the vanishing gradient problem on sigmoid activation. *Progress in Artificial Intelligence*, 9(4):351–360.

Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

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

Song, H. F., Yang, G. R., and Wang, X.-J. (2016). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology*, 12(2):e1004792.

Strata, P. and Harvey, R. (1999). Dale's principle. *Brain research bulletin*, 50(5-6):349–350.

Szandała, T. (2020). Review and comparison of commonly used activation functions for deep neural networks. In *Bio-inspired neurocomputing*, pages 203–224. Springer.

Turrigiano, G. (2011). Too many cooks? intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement. *Annual review of neuroscience*, 34(1):89–103.

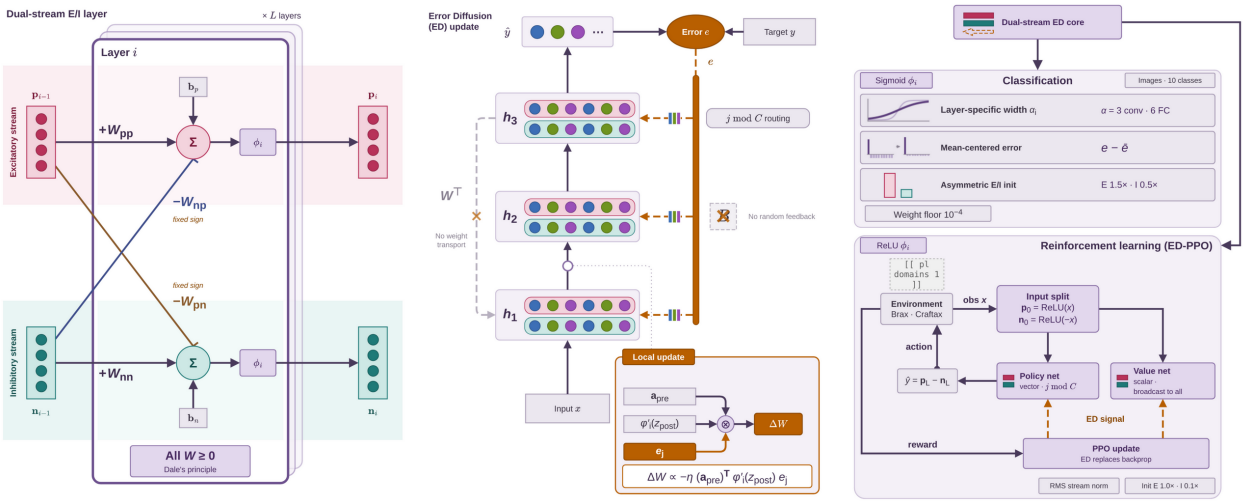
Whittington, J. C. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262.

Error Diffusion ALIFE Paper (pages 9–10 of 10)

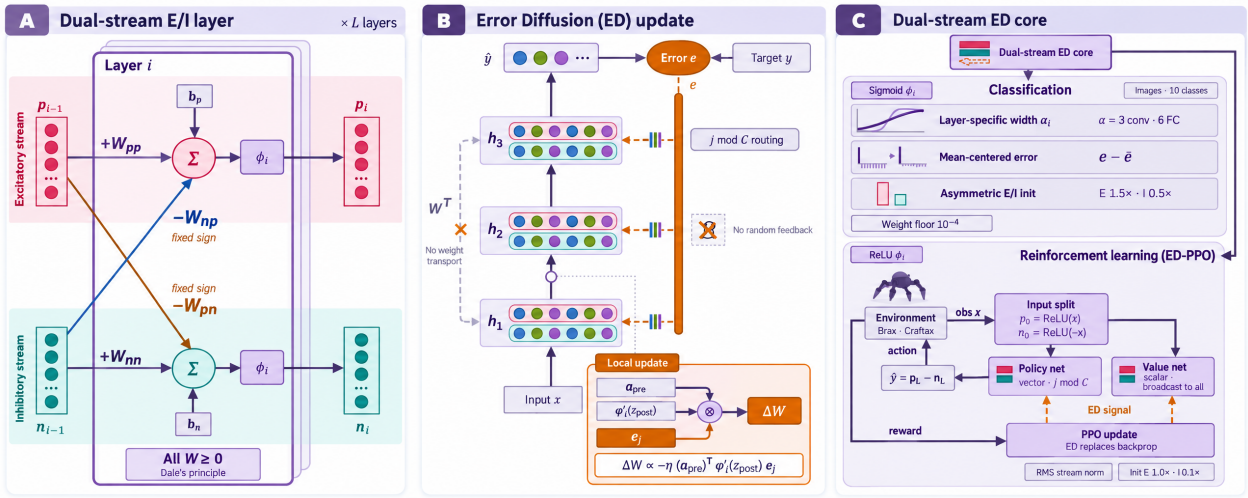
D. Error Diffusion ALIFE Paper Details

D.1. Error Diffusion ALIFE Paper: Agent Draft to Submission Diff

As this paper was produced with a development version of THE AGENTIC SCIENTIST, its conceptual figure was originally generated with Nano Banana Pro (Google DeepMind, 2025), rather than with AUTOILLUSTRATE. Since Nano Banana Pro outputs a raster image without an editable source, the human authors could not correct its factual inaccuracies in place: before submission, they kept the right panel, which was correct, and manually redrew the left and middle panels. The current version of THE AGENTIC SCIENTIST uses AUTOILLUSTRATE by default for conceptual figures (Sections 3.5 and 3.6), which produces more faithful figures than the Nano Banana Pro-based PaperBanana pipeline (Zhu et al., 2026a) (Section 7). For reference, we applied the current version of AUTOILLUSTRATE to the agent draft of the ALIFE paper, before any human edits and adjustments. Figure 20 shows the resulting editable `draw.io` diagram and the enhanced figure derived from it.



(a) Editable `draw.io` diagram.



(b) Enhanced figure derived from the editable diagram.

Figure 20 | AUTOILLUSTRATE applied to the agent draft of the Error Diffusion ALIFE paper, before human edits: the editable diagram **(a)** and its corresponding enhanced figure **(b)**. The figure in the submitted and camera-ready versions of the ALIFE paper was instead generated with Nano Banana Pro as a non-editable raster image; the human authors kept its correct right panel and manually redrew the left and middle panels to correct their inaccuracies. The prompt given to AUTOILLUSTRATE was: "Overview of the dual-stream Error Diffusion framework. **Left:** The excitatory/inhibitory architecture maintains separate streams (**p**, **n**) with four non-negative weight matrices per layer, enforcing Dale's principle structurally. **Center:** The Error Diffusion update broadcasts output error directly to hidden layers without weight transport or random feedback matrices. **Right:** The shared architecture is applied to classification (with domain-specific sigmoid widths, mean-centered error, and asymmetric initialization) and reinforcement learning (with ReLU activations and PPO integration)."

Diffusing Blame: Task-Dependent Credit Assignment ~~in~~ in Biologically Plausible Dual-Stream ~~Error Diffusion NetworksUnder Dale's Principle~~Networks

Yutaro Yamada, Luca Grillotti, Rujikorn Charakorn, Sebastian Risi, David Ha, Robert Tjarko Lange

Sakana AI, Tokyo, Japan
yutaro.yamada.y@gmail.com, robert@sakana.ai

Abstract

Biological neural circuits obey Dale's principle: each neuron's synapses are uniformly excitatory or inhibitory. Artificial networks that respect this constraint must coordinate separate excitatory and inhibitory populations, fundamentally changing how credit is assigned during learning. Several biologically plausible learning rules avoid backpropagation's ~~weight-transport requirement, yet none has been shown to scale~~ weight-transport requirement, but achieving strong performance under Dale's principle ~~across both supervised classification and reinforcement learning~~beyond MNIST has been difficult. Error Diffusion (ED) ~~is a local learning rule that routes a single global error sign~~ was originally proposed in a dual-stream excitatory/inhibitory architecture, where learning is driven by routing global error signals to all layers without transporting ~~weights or using~~ transposed forward weights or relying on random feedback matrices. ~~Here,~~ Whether such a rule can scale under Dale's principle across both supervised classification and reinforcement learning remains unknown. Here, we show that a dual-stream excitatory/inhibitory architecture trained with ED achieves 96.7% on MNIST and establishes a robust 61.7% baseline on CIFAR-10~~for classification, and achieves comparable or superior mean reward to backpropagation-based PPO on continuous-control RL benchmarks significantly exceeding it on Humanoid ($p = 0.007$, Welch t-test)~~ all while maintaining non-negative weights, demonstrating that competitive representation learning is possible even when strictly enforcing Dale's principle. For the classification setting, we introduce three domain-specific ~~innovations: layer-specific~~ innovations: layer-specific sigmoid widths, mean-centered error signals, and asymmetric ~~initialization, and~~initialization. An ablation analysis reveals that their relative importance *reverses* between MNIST and CIFAR-10, exposing task-dependent credit-assignment bottlenecks that are invisible to single-benchmark evaluation. In ~~RL, the same~~ reinforcement learning, by contrast, we retain the dual-stream architecture ~~integrated with PPO achieves competitive rewards without these classification-specific stabilizers~~and multidimensional error routing but relax the strict non-negativity constraint: in this regime, ED, integrated effectively with Proximal Policy Optimization (PPO), achieves competitive performance, demonstrating that the core architecture generalizes across domains even as the optimal auxiliary mechanisms shift.

Introduction

In biological neural circuits, the separation of excitatory and inhibitory neuronal populations is a fundamental organizational principle, formalized as Dale's principle: each neuron releases the same neurotransmitter at all of its synapses (Dale, 1935; Eccles, 1976; Strata and Harvey, 1999). This constraint shapes cortical computation through balanced excitation and inhibition, and its disruption is implicated in neurological disorders (Markram et al., 2004). Artificial neural networks, by contrast, freely assign positive and negative weights to any connection, a simplification that enables efficient credit assignment via backpropagation but bears little resemblance to biological learning.

Training deep networks with backpropagation requires the backward pass to use exact transposes of forward weight ~~matrices, the~~matrices, the *weight transport problem*~~, which~~, which lacks biological support (Crick, 1989; Bengio et al., 2015). Several alternatives address this concern. Feedback Alignment (FA) replaces backward weights with fixed random matrices (Lillicrap et al., 2016), Direct Feedback Alignment (DFA) routes output errors directly to each hidden layer (Nøkland, 2016), and predictive coding frameworks approximate backpropagation through local Hebbian updates (Whittington and Bogacz, 2017; Sacramento et al., 2018). While these methods avoid weight transport, none ~~enforces~~ enforce Dale's principle: all allow arbitrary sign weights. Error Diffusion (ED), originally proposed by (Kaneko, 2000), is a local learning rule in which weight updates depend on presynaptic activity, a post-synaptic activation derivative, and a single global error sign ~~(Dellaferrera and Kreiman, 2022)~~(Fujita, 2026). This locality makes ED naturally compatible with biological constraints, but prior work demonstrated it only on binary classification.

The central question we address is whether a biologically plausible architecture that enforces Dale's principle can achieve competitive performance across both supervised classification and reinforcement learning, and what the task-dependent importance of its components reveals about credit assignment under biological constraints. We make three contributions:

1. A dual-stream excitatory/inhibitory ED architecture that maintains non-negative weights across classification (MNIST, CIFAR-10) and reinforcement learning (Brax locomotion tasks, Craftax).
2. For the classification setting, three domain-specific ~~innovations: layer-specific~~ innovations: layer-specific sigmoid widths, mean-centered error, and asymmetric ~~initialization, whose~~initialization, whose cross-task ablation reveals that credit-assignment bottlenecks shift qualitatively with task difficulty.
3. A cross-domain evaluation showing that ED-PPO ~~significantly exceeds~~ achieves comparable returns to backpropagation-based PPO on ~~Humanoid ($p = 0.007$) and achieves comparable rewards on Ant and~~ Ant, Humanoid and HalfCheetah, while DFA fails on the most complex task (Craftax, $p = 0.002$ vs ED-PPO), demonstrating that the structured error routing of ED generalizes more robustly than random feedback pathways.

Related Work

Biologically plausible learning rules. The weight transport problem has motivated a family of alternatives to backpropagation (Crick, 1989; Bengio et al., 2015). Feedback Alignment uses fixed random backward weights (Lillicrap et al., 2016), and Liao et al. (2016) showed that approximate weight symmetry can support learning in deeper networks. DFA projects output errors directly to hidden layers (Nøkland, 2016), and Launay et al. (2020) demonstrated that DFA scales to modern architectures, including transformers and convnets. Difference target propagation computes layer-wise learning targets instead of gradients (Lee et al., 2015). Predictive coding provides a complementary framework through local Hebbian updates (Whittington and Bogacz, 2017), and dendritic cortical microcircuit models approximate backpropagation with segregated dendrites (Sacramento et al., 2018). Bartunov et al. (2018) found that biologically motivated algorithms generally struggle to scale beyond simple benchmarks, and Richards et al. (2019) argued that biological plausibility and computational performance need not be in tension if the right architectural principles are identified. Filipovich et al. (2022) demonstrated DFA on photonic hardware, showing that bioplausible rules can enable novel computing substrates.

Dale's principle in neural networks. Song et al. (2016) introduced a framework for training excitatory-inhibitory recurrent networks with Dale's principle, re-parameterizing weights as $W = |W^+| \odot M$ where M is a fixed sign mask. Grossberg (1987) explored competitive learning in networks with separate excitatory and inhibitory interactions. Our approach differs in using separate excitatory and inhibitory populations (dual-stream) rather than a sign mask, which allows independent representations in each stream. Error Diffusion (Dellaferrera and Kreiman, 2022) avoids both weight transport and random feedback, motivating our multi-class and RL extensions.

Reinforcement learning without backpropagation. Evolution Strategies (ES) provide a gradient-free alternative to policy gradient methods (Salimans et al., 2017), but scale poorly with parameter count. Proximal Policy Optimization (PPO) is a widely used policy gradient method (Schulman et al., 2017) that typically relies on backpropagation for gradient computation. Biologically plausible alternatives for RL credit assignment have been proposed—reward-modulated Hebbian rules can solve decision-making tasks by gating synaptic updates with a global reward signal (Pfeiffer et al., 2010), and Izhikevich (2007) showed that linking spike-timing-dependent plasticity with dopamine signaling addresses the distal reward problem—but these methods have not been demonstrated with deep networks on continuous-control benchmarks. We replace backpropagation in PPO's gradient computation with Error Diffusion, evaluating on Brax physics tasks (Freeman et al., 2021) and Craftax (Matthews et al., 2024).

Method

Dual-Stream Architecture~~(Shared Core)~~

We adopt the original Error Diffusion formulation proposed by (Kaneko, 2000). To enforce Dale's ~~principle—all weights non-negative—we~~ principle (all weights being non-negative), we split each layer into an excitatory (**p**) and inhibitory (**n**) stream —The forward pass for layer i computes:

$$\mathbf{p}_i = \phi_i(+\mathbf{p}_{i-1}W_{pp} - \mathbf{n}_{i-1}W_{np} + \mathbf{b}_p)$$
$$\mathbf{n}_i = \phi_i(+\mathbf{n}_{i-1}W_{nn} - \mathbf{p}_{i-1}W_{pn} + \mathbf{b}_n)$$

where all weight matrices $W_{pp}, W_{np}, W_{nn}, W_{pn} \geq 0$ element-wise, and ϕ_i is a layer-specific activation function~~(sigmoid for classification, ReLU for RL)~~. The negation signs in front of W_{np} and W_{pn} are structural (hardcoded), ensuring that cross-stream connections are inhibitory while all learnable parameters remain non-negative. This dual-stream design requires four weight sub-matrices per layer, resulting in $\sim$4$\times$ more parameters than an unconstrained single-stream network of the same width (e.g., $\sim$32M vs $\sim$8M for DFA on the same architecture).

Error Diffusion Update Rule

The weight update follows the ED principle: each weight is adjusted proportionally to the presynaptic activity, the post-synaptic activation derivative, and a global error signal:

$$\Delta W \propto -\eta \cdot \mathbf{a}_{\text{pre}}^{\top} \cdot \phi_i'(z_{\text{post}}) \cdot e_j \qquad (1)$$

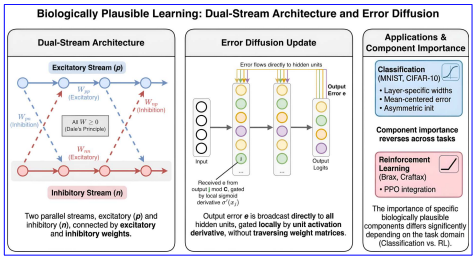


Figure 1: Overview of the dual-stream Error Diffusion framework. **Left:** The excitatory/inhibitory architecture maintains separate streams (**p**, **n**) with four non-negative weight matrices per layer, enforcing Dale's principle structurally. **Center:** The Error Diffusion update broadcasts output error directly to hidden layers without weight transport or random feedback matrices. **Right:** The shared architecture is applied to classification (with domain-specific sigmoid widths, mean-centered error, and asymmetric initialization) and reinforcement learning (with ReLU activations and PPO integration).

where ϕ_i' is the activation derivative. While the original formulation is for binary classification (Kaneko, 2000; Fujita, 2026), we extend ED from the binary to the multi-output dimension in the following way. The error signal is computed from the output layer: each hidden unit j receives the error from output unit $j \bmod C$ (where C is the number of output units), providing coarse credit assignment without requiring per-synapse error gradients, weight transport, or random feedback matrices. This modular routing is deterministic and fixed, in contrast to DFA, which uses random projection matrices that are also fixed but lack structured correspondence to the output.

Classification-Specific Innovations

For the classification setting (MNIST, CIFAR-10), we introduce three domain-specific innovations that address failure modes specific to multi-class classification under Dale's principle. These three innovations are specific to the classification setting and are not used in the RL extension.

Layer-specific sigmoid widths. In the classification architecture, $\phi_i(z) = 1/(1 + e^{-z/\alpha_i})$ with layer-specific width α_i. The vanishing gradient problem in sigmoid networks is well documented (Roodschild et al., 2020; Ding et al., 2018), and various parameterized activations have been proposed (Szandała, 2020; Hammad, 2024). Since the sigmoid derivative directly gates the error signal in ED (Eq. 1), gradient attenuation is especially severe: post-hoc analysis reveals a $25\times$ decay from output to first hidden layer. Wider sigmoids (larger α_i) maintain larger derivatives, preventing premature saturation. We set $\alpha = 3.0$ for convolutional layers and $\alpha = 6.0$ for fully connected layers.

Mean-centered error signals. In 10-class classification with one-hot targets, the raw error $e = y - \hat{y}$ contains one positive and nine negative entries per sample. We subtract the per-class mean: $e_{\text{centered}} = (y - \hat{y}) - \text{mean}(y - \hat{y})$. This rebalances the positive and negative error components, preventing the update from being dominated by suppressing incorrect classes.

Asymmetric E/I initialization. ~~We initialize~~ For feedforward layers, weight initialization samples weight parameters uniformly between 0 and 1. Afterwards, these parameters are additionally scaled by the number of incoming units. Finally, excitatory weights (W_{pp}, W_{nn}) ~~at~~ are scaled $1.5\times$ ~~here scale~~ and inhibitory weights (W_{np}, W_{pn}) ~~at~~ $0.5\times$, creating an initial E/I ratio of approximately 3:1. The output layer uses symmetric initialization (1:1) to avoid output saturation.

Figure 1 summarizes the error diffusion training procedure and methodological contributions.

RL Extension: ED-PPO

For reinforcement learning, we integrate the dual-stream ED architecture into PPO (Schulman et al., 2017). The policy and value networks both use the dual-stream architecture with ReLU activations and RMS stream normalization between layers. Input observations are split into positive and negative streams via $\mathbf{p}_0 = \text{ReLU}(x)$, $\mathbf{n}_0 = \text{ReLU}(-x)$, and the output is computed as $\hat{y} = \mathbf{p}_L - \mathbf{n}_L$. ED provides the gradient signal in place of backpropagation: for the policy network (vector output), the same modular routing applies; for the value network (scalar output), the error broadcasts directly to all hidden units. Weights are initialized non-negative via $|W| \sim |\mathcal{N}(0, \sigma^2)|$ with asymmetric scaling (excitatory at $1.0\times$, inhibitory at $0.1\times$).

Experimental Protocol

Classification. We evaluate six variants on MNIST and CIFAR-10: (1) the **proposed ED** with all three innovations, (2) a **DFA baseline** using fixed random feedback matrices without Dale's constraints, (3) the **seed ED** (the dual-stream architecture without any of the three innovations, i.e., using uniform sigmoid width $\alpha = 1.0$, raw error signals, and symmetric initialization), and three ablations removing one innovation each: (4) **no mean-centered error**, (5) **symmetric init**, and (6) **uniform sigmoid width**. Each variant is trained for 250 epochs with five random seeds per task (60 runs total). ~~In the classification setting~~For classification, weights are clamped to a floor of 10^{-4} after each update to enforce Dale's principle.

Reinforcement learning. We compare ED-PPO against BP-PPO (standard backpropagation) and ES (Salimans et al., 2017) on three Brax (Freeman et al., 2021) locomotion tasks (Ant, HalfCheetah, Humanoid) and Craftax (Matthews et al., 2024). On Craftax, we additionally compare against DFA-PPO. Each algorithm is evaluated with five random seeds per environment. We report the final episode reward (the reward at the last evaluation checkpoint) averaged over seeds; statistical comparisons use Welch's t-test with $\alpha = 0.05$. The RL experiments use a separate training and evaluation pipeline from the classification experiments, with both pipelines sharing the same dual-stream ED architecture but differing in activation functions, normalization, and optimization procedure as described above.

Results

Classification: Main Comparison

~~Table 1~~Figure 2 summarizes accuracy across all six classification variants. The proposed ED achieves $96.7 \pm 0.1\%$ on MNIST and $61.7 \pm 0.7\%$ on CIFAR-10, a substantial improvement over seed ED ($50.4 \pm 9.8\%$ and $11.6 \pm 2.2\%$).

Table 1: ~~Classification accuracy (%, mean ± std, 5 seeds). Bold: best Dale-compliant. †Violates Dale's principle.~~

~~Variant~~	~~MNIST~~	~~CIFAR-10~~
~~DFA†~~	~~97.6 ± 0.1~~	~~69.1 ± 1.9~~
~~**Proposed ED**~~	~~**96.7 ± 0.1**~~	~~**61.7 ± 0.7**~~
~~No MCE~~	~~96.4 ± 0.1~~	~~13.8 ± 8.6~~
~~Sym. Init~~	~~96.7 ± 0.1~~	~~58.2 ± 1.8~~
~~Unif. Width~~	~~25.3 ± 8.9~~	~~46.6 ± 10.8~~
~~Seed ED~~	~~50.4 ± 9.8~~	~~11.6 ± 2.2~~

Table 2: Ablation effect sizes (Δ = proposed ED minus ablation, in ~~percentage~~points). Negative values indicate accuracy loss.

Removed Component	MNIST Δ	CIFAR-10 Δ
Mean-Centered Error	-0.3	-47.9
Layer-Specific Widths	-71.4	-15.1
Asymmetric Init	$+0.0$	-5.5

DFA achieves higher accuracy on both tasks (97.6% and 69.1%) but violates Dale's principle, requiring $\sim$2.8M negative weights. The accuracy gap between ED and DFA widens from 0.9 pp on MNIST to 7.4 pp on CIFAR-10, suggesting that the cost of maintaining non-negative weights grows with task difficulty. We note that 62% test accuracy on CIFAR-10 is still far from competitive compared to traditional gradient-based methods. That being said, this is the first time ED has been successfully applied to convolutional neural networks. Previously, Fujita (2026) obtained $\sim$ 55.2% test accuracy on CIFAR-10 using a purely feedforward MLP with flattening.

Classification: Cross-Task Ablation Reversal

In the classification setting, ablation analysis reveals that the importance hierarchy of the three innovations reverses between tasks (Figure 2, Table 2).

MNIST. Removing layer-specific sigmoid widths is catastrophic (96.7% $\rightarrow$ 25.3%, $\Delta = -71.4$ pp), collapsing accuracy to near-chance. Removing mean-centered error causes only $\Delta = -0.3$ pp, and symmetric initialization has no measurable effect ($\Delta = +0.0$ pp). Gradient flow regulation is the sole bottleneck on this easier task.

CIFAR-10. The hierarchy reverses. Removing mean-centered error is now the most destructive ablation (61.7% $\rightarrow$ 13.8%, $\Delta = -47.9$ pp), causing four of five seeds to collapse. Uniform sigmoid width causes $\Delta = -15.1$ pp, and symmetric initialization causes $\Delta = -5.5$ pp. All three innovations contribute, but their relative ordering changes fundamentally.

Table 3: ~~RL final reward (mean ± std, 5 seeds). **Bold**: best per environment. *Significant (Welch t-test, $p < 0.05$) vs second best. ED-PPO uses the dual-stream ED architecture; BP-PPO uses standard backpropagation.~~

~~Environment~~	~~ED-PPO~~	~~BP-PPO~~	~~ES/DFA~~
~~Ant~~	~~7616 ± 2031~~	~~6367 ± 85~~	~~4868 ± 1034~~
~~HalfCheetah~~	~~3498 ± 1703~~	~~4515 ± 101~~	~~4705 ± 788~~
~~Humanoid~~	~~9804 ± 1144*~~	~~6073 ± 1303~~	~~5656 ± 697~~
~~Craftax~~	~~23.0 ± 1.5~~	~~27.0 ± 0.9*~~	~~19.8 ± 1.5~~

Interpretation. This reversal reflects qualitatively different credit-assignment bottlenecks. MNIST's well-separated features allow learning even without error centering, but without wide sigmoids, the derivatives vanish entirely. On CIFAR-10, higher inter-class similarity makes the 9:1 error imbalance overwhelming: mean centering becomes essential to prevent uniform output suppression. The reversal demonstrates that evaluating biologically plausible methods on a single benchmark may obscure critical design trade-offs.

Reinforcement Learning: Cross-Domain Generalization

~~Table 3 and Figure 3~~ Figures 3 and 4 present ED-PPO performance across four RL environments. ED-PPO ~~significantly exceeds~~ performs on par with BP-PPO on Humanoid (9804 ± 1144 vs ~~6073 ± 1303; Welch $t = 3.65$, $p = 0.007$~~8478 ± 3252). On Ant, ED-PPO achieves a higher mean (7616 ± 2031 vs ~~6367 ± 85~~6740 ± 1781) but the difference is not statistically significant ($p = 0.241$~~; $p = 0.214$~~) due to high variance. On HalfCheetah, ED-PPO achieves a ~~lower mean~~ similar score (3498 ± 1703 vs ~~4515 ± 101; $p = 0.253$~~3520 ± 485), with one of five seeds collapsing to low reward. On Craftax, BP-PPO significantly leads (27.0 ± 0.9 vs 23.0 ± 1.5; $p = 0.002$). ES consistently ranks below both PPO variants on all Brax tasks. Notably, DFA-PPO on Craftax achieves only 19.8 ± 1.5, significantly below ED-PPO ($p = 0.010$), demonstrating that random feedback pathways that suffice for simpler tasks fail on complex, open-ended environments.

ED-PPO exhibits substantially higher variance than BP-PPO across all environments (e.g., Ant std: 2031 vs 85), suggesting that the ED gradient signal introduces additional stochasticity. ~~The only statistically significant advantage is on Humanoid ($+2831$, $p = 0.007$), where ED-PPO consistently outperforms across all seeds.~~ On HalfCheetah, one ED-PPO seed collapses to low reward (943), inflating variance and reducing the mean below BP-PPO. The Craftax shortfall (-4.0, $p = 0.002$) and higher variance across environments suggest that the coarse modular error routing of ED may be less reliable for tasks requiring fine-grained temporal credit assignment or stable convergence.

DFA Across Domains

DFA provides an informative comparison point across both classification and RL. In classification, DFA achieves the highest accuracy (97.6% MNIST, 69.1% CIFAR-10) but violates Dale's principle. On Craftax, DFA-PPO is the weakest method (19.8 vs BP-PPO 27.0, ED-PPO 23.0), demonstrating that random feedback pathways that work for supervised learning can fail for complex RL. This cross-domain pattern—DFA strong on classification, weak on complex RL—parallels the finding that method components assume different importance under different task demands.

Mechanistic Evidence

Gradient attenuation. Post-hoc analysis of training dynamics on CIFAR-10 (Figure 5a) reveals gradient magnitudes dropping from 6.6×10^{-3} at the output layer to 2.7×10^{-4} at the first hidden layer—a $25\times$ attenuation. This pattern is visible from epoch 3, indicating a structural property of sigmoid-gated error diffusion that motivates layer-specific widths. Per-layer learning rate multipliers ($3.0\times$ early, $0.5\times$ output) partially compensate but cannot substitute for wider sigmoid derivatives. The near-zero generalization gap (0.98%) despite ∼32M nominal parameters is consistent with underfitting rather than overfitting, but the cause is ambiguous: the 37.3% of weights at floor (10^{-4}) reduces effective capacity well below the nominal parameter count, so the limitation may reflect constrained capacity rather than (or in addition to) credit assignment quality.

Emergent E/I balance. Weight-level E/I ratios (Figure 5b) reveal that training drives the 3:1 asymmetric initialization toward near-balanced ratios (∼1.0) in hidden layers. A depth-dependent gradient emerges: the first layer reaches near-perfect balance (E/I ≈ 1.03), the second becomes slightly inhibitory-dominant (0.90), and the third develops the strongest inhibitory bias (0.81). This increasing inhibition with depth is loosely analogous to biological observations of inhibitory interneuron density increasing in higher cortical areas (Markram et al., 2004), though weight-level E/I ratios are an indirect proxy for biological E/I balance, which involves cell counts, firing rates, and synaptic strengths jointly. The convergence toward balance explains the task-dependent importance of asymmetric initialization: on MNIST, balance is reached quickly regardless of initial conditions ($\Delta = +0.0$ pp); on CIFAR-10, the asymmetric head start prevents early instability ($\Delta = -5.6$ pp).

Implicit sparsity. The non-negative weight floor induces substantial implicit sparsity: 37.3% of weights reach the floor (10^{-4}) after training. Inhibitory (cross-stream) FC connections are pruned most aggressively (up to 68.8% at the floor), compared to 26–49% for excitatory connections. Convolutional layers are much less affected (<1% to 18%).

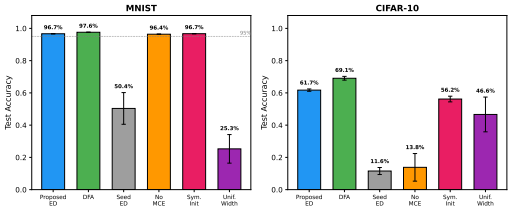


Figure 2: Classification accuracy across all six variants on MNIST (left) and CIFAR-10 (right). Error bars: ±1 std over 5 seeds. The ablation importance hierarchy reverses between tasks: layer-specific widths dominate on MNIST, mean-centered error dominates on CIFAR-10.

This asymmetric pruning suggests that Dale's principle introduces a structured capacity bottleneck that preferentially eliminates inhibitory connections in deeper layers.

Discussion

Summary. The central finding of this work is that the dual-stream ED architecture is viable across both classification and reinforcement learning, but the auxiliary mechanisms required for competitive performance are domain-dependent. In classification, mean-centered error and layer-specific sigmoid widths are critical stabilizers whose relative importance reverses between MNIST and CIFAR-10. In RL, the same dual-stream architecture with ReLU activations and RMS ~~normalization without~~ normalization, without sigmoid widths or mean ~~centering achieves~~ centering, achieves competitive performance on locomotion tasks, ~~significantly exceeding backpropagation based PPO on Humanoid while~~ showing comparable but ~~more~~ variable rewards on Ant, Humanoid, and HalfCheetah.

~~This domain dependence has implications for biological plausibility research.~~

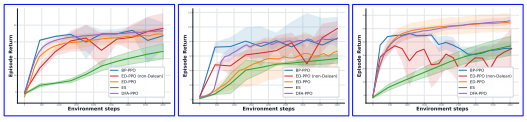
Implications for biological plausibility research. Single-benchmark evaluations can produce misleading conclusions about which architectural features matter. The ablation reversal between MNIST and CIFAR-10 demonstrates that a component critical for one task can be negligible for another, and the DFA cross-domain pattern (strong on classification, weak on complex RL) reinforces this point. ~~Future work on biologically plausible learning should evaluate across task domains, not just task difficulty levels.~~

The emergent E/I balance ~~convergence from convergence, from~~ 3:1 toward ∼1:1 with a depth-dependent inhibitory ~~gradient provides~~ gradient, provides a connection to biological self-organization. Cortical circuits are known to develop balanced E/I ratios during maturation through intrinsic and synaptic homeostatic mechanisms (Markram et al., 2004; Turrigiano, 2011), and our results suggest that a similar homeostatic process can emerge from gradient-driven learning under Dale's principle constraints, without explicit balance-enforcing mechanisms.

The success of ED-PPO on locomotion tasks despite using ReLU rather than sigmoid activations raises an interesting question about the relationship between activation function choice and credit assignment quality. In classification, the sigmoid derivative explicitly gates the error signal, making the width parameter critical. In RL, ReLU's unbounded positive derivative may provide a natural gradient pathway that avoids the attenuation problem without requiring layer-specific tuning. This activation-dependent difference in error flow may partially explain why the classification-specific innovations are unnecessary in the RL setting.

~~Several limitations qualify these findings.~~

Limitations. The classification innovations (sigmoid widths, MCE, asymmetric init) are specific to the classification setting; we do not claim they transfer to RL. ED-PPO shows substantially higher variance than BP-PPO across all environments, suggesting that the coarse error routing introduces stochasticity that may limit reliability. The Craftax



(a) Ant (b) Humanoid (c) HalfCheetah

Figure 3: ~~Final reward across four RL environments comparing ED-PPO, BP-PPO, and ES (or DFA on Craftax). Error bars: ±1 std over 5 seeds. ED-PPO significantly exceeds BP-PPO only on Humanoid ($p = 0.007$); differences on Ant and HalfCheetah are not statistically significant.~~ Final reward across three RL environments comparing ED-PPO, BP-PPO, and ES. Error bars indicate ±1 standard deviation over 5 seeds. The comparison between ED-PPO and BP-PPO on Humanoid is not statistically significant ($p = 0.42$); differences on Ant and HalfCheetah are not statistically significant.

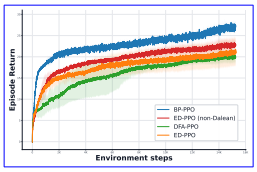
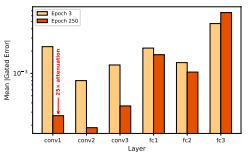


Figure 4: Episode return on Craftax comparing ED-PPO, BP-PPO, and DFA. Error bars: ±1 std over 5 seeds.

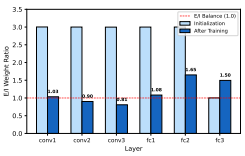
shortfall (-4.0 reward vs BP-PPO) indicates that ED's modular credit assignment may be insufficient for tasks requiring fine-grained temporal reasoning. The accuracy gap between ED and DFA on classification (-0.9 to -7.4 pp) quantifies the current cost of enforcing Dale's principle. On Craftax, DFA underperforms ED-PPO, suggesting that the cost of Dale's principle is task-dependent. Finally, the dual-stream architecture's ∼4× parameter overhead may itself limit effective capacity, particularly on classification where 37.3% of weights are pruned to the floor.

From the perspective of adaptive computation, the emergent behaviors we observe—E/I balance self-organization, asymmetric pathway pruning, and task-dependent component criticality—suggest that biologically constrained architectures may develop internal regulatory mechanisms analogous to those found in living systems. ~~Whether these parallels reflect shared computational principles or merely superficial similarities remains an open question for future investigation.~~

Future Work. Several directions follow naturally from the results presented here. First, the non-negativity constraint imposed by Dale's principle may translate into concrete hardware advantages: sign-free arithmetic simplifies multiply-accumulate circuitry and could enable more efficient analog or neuromorphic substrates, complementing prior demonstrations of bioplausible rules on photonic hardware (Filipovich et al., 2022). Second, the implicit sparsity we observe suggests that Dale-compliant training may yield model compression "for free," since weights bounded below at zero cannot recover once suppressed, turning the floor into a natural pruning mechanism that could be exploited with structured sparsity kernels at inference time. Third, we conjecture that the dual-stream architecture may be unusually well-suited to continual and open-ended learning settings, where catastrophic forgetting is a central obstacle: the dedicated inhibitory stream provides a structural mechanism for dampening large gradient excursions, and the sign constraint prevents the kind of unconstrained weight sign flips that are thought to contribute to representational overwriting. Evaluating ED on sequential task streams and lifelong RL benchmarks is a natural next step. Fourth, the segregation of computation into excitatory "amplifiers" and inhibitory "suppressors" offers a potential interpretability handle that standard networks lack: when a model makes an error, one can, in principle trace whether the mistake originated from a feature being falsely amplified by the excitatory stream or insufficiently suppressed by the inhibitory stream, providing a more mechanistic form of attribution than gradient-based saliency on sign-unconstrained networks. Finally, closing the accuracy gap to DFA and BP-PPO on the hardest tasks, whether through richer error-routing schemes, learned (rather than fixed modular) output-to-hidden projections, or hybrid activation functions that preserve sigmoid locality while mitigating attenuation, remains the most direct path toward making Dale-compliant learning competitive with unconstrained backpropagation.



(a) Per-layer gradient magnitudes at epochs 3 and 250 on CIFAR-10 (log scale). The 25× attenuation from output to first hidden layer motivates layer-specific sigmoid widths.



(b) E/I weight ratio at init (3:1 for hidden layers) vs. after 250 epochs. Hidden layers converge toward balance; deeper layers become inhibitory-dominant.

Figure 5: Post-hoc mechanistic analyses of the proposed ED model on CIFAR-10. (a) Gradient attenuation across layers. (b) Emergent E/I balance from asymmetric initialization toward biological-like equilibrium.

Conclusion

We demonstrated that a dual-stream excitatory/inhibitory architecture trained with Error Diffusion achieves competitive performance across both supervised classification and reinforcement learning while maintaining non-negative weights consistent with Dale's principle. In the classification setting, three domain-specific innovations enable scaling from binary to 10-class problems, and their cross-task ablation reversal reveals that credit-assignment bottlenecks shift qualitatively with task difficulty. In reinforcement learning, the same core architecture integrated with PPO ~~significantly exceeds backpropagation on Humanoid and~~ achieves comparable rewards ~~on other~~ to backpropagation-based PPO on locomotion tasks, without requiring the classification-specific stabilizers, though with substantially higher variance. The cross-domain ~~pattern—competitive pattern,~~ competitive performance with domain-dependent auxiliary ~~mechanisms—suggests~~ mechanisms, suggests that biologically plausible learning rules need not be monolithic: a shared architectural core can support diverse task demands when augmented with appropriate domain-specific components. The accuracy gap relative to DFA and BP-PPO on the hardest tasks quantifies the current cost of Dale's principle compliance and provides concrete benchmarks for future work on narrowing this gap while preserving biological fidelity.

References

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Dale, H. (1935). Pharmacology and nerve-endings. *Proceedings of the Royal Society of Medicine*.

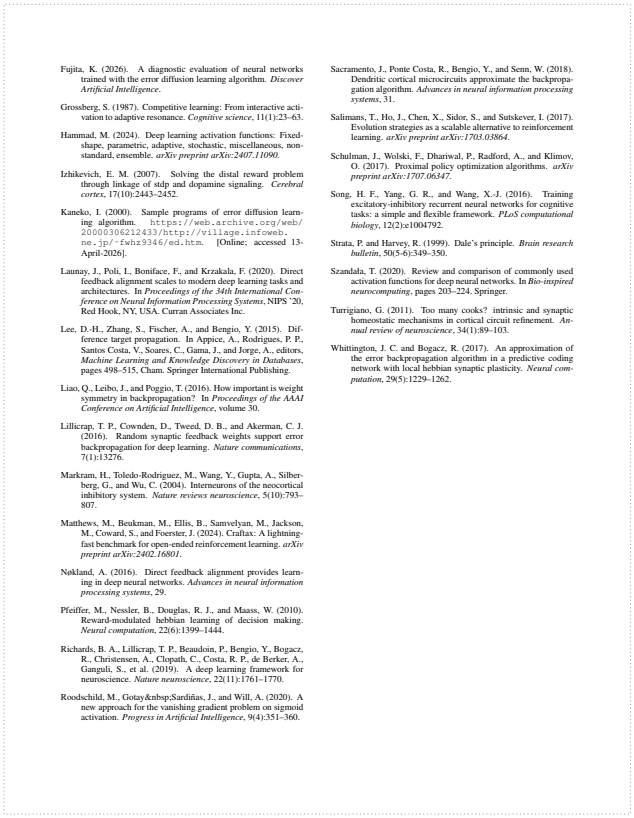
Dellaferrera, G. and Kreiman, G. (2022). Error-driven input modulation: solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pages 4937–4955. PMLR.

Ding, B., Qian, H., and Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841.

Eccles, J. C. (1976). From electrical to chemical transmission in the central nervous system: the closing address of the sir henry dale centennial symposium cambridge, 19 september 1975. *Notes and records of the Royal Society of London*, 30(2):219–230.

Filipovich, M. J., Guo, Z., Al-Qadasi, M., Marquez, B. A., Morison, H. D., Sorger, V. J., Prucnal, P. R., Shekhar, S., and Shastri, B. J. (2022). Silicon photonic architecture for training deep neural networks with direct feedback alignment. *Optica*, 9(12):1323–1332.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax-a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*.



Fujita, K. (2026). A diagnostic evaluation of neural networks trained with the error diffusion learning algorithm. *Discover Artificial Intelligence*.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63.

Hammad, M. (2024). Deep learning activation functions: Fixed-shape, parametric, adaptive, stochastic, miscellaneous, non-standard, ensemble. *arXiv preprint arXiv:2407.11090*.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452.

Kaneko, I. (2000). Sample programs of error diffusion learning algorithm. `https://web.archive.org/web/20000306212433/http://village.infoweb.ne.jp/~fwhz9346/ed.htm`. [Online; accessed 13-April-2026].

Launay, J., Poli, I., Boniface, F., and Krzakala, F. (2020). Direct feedback alignment scales to modern deep learning tasks and architectures. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. In Appice, A., Rodrigues, P. P., Santos Costa, V., Soares, C., Gama, J., and Jorge, A., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 498–515, Cham. Springer International Publishing.

Liao, Q., Leibo, J., and Poggio, T. (2016). How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276.

Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., and Wu, C. (2004). Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience*, 5(10):793–807.

Matthews, M., Beukman, M., Ellis, B., Samvelyan, M., Jackson, M., Coward, S., and Foerster, J. (2024). Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29.

Pfeiffer, M., Nessler, B., Douglas, R. J., and Maass, W. (2010). Reward-modulated hebbian learning of decision making. *Neural computation*, 22(6):1399–1444.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., et al. (2019). A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770.

Roodschild, M., Gotay Sardiñas, J., and Will, A. (2020). A new approach for the vanishing gradient problem on sigmoid activation. *Progress in Artificial Intelligence*, 9(4):351–360.

Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

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

Song, H. F., Yang, G. R., and Wang, X.-J. (2016). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology*, 12(2):e1004792.

Strata, P. and Harvey, R. (1999). Dale's principle. *Brain research bulletin*, 50(5-6):349–350.

Szandała, T. (2020). Review and comparison of commonly used activation functions for deep neural networks. In *Bio-inspired neurocomputing*, pages 203–224. Springer.

Turrigiano, G. (2011). Too many cooks? intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement. *Annual review of neuroscience*, 34(1):89–103.

Whittington, J. C. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262.

Error Diffusion ALIFE Paper: agent draft to submission diff (page 9 of 9)

D.2. Error Diffusion ALIFE Paper: Submission to Camera-Ready Diff

Diffusing Blame: Task-Dependent Credit Assignment in Biologically Plausible Dual-Stream Networks

Yutaro Yamada, Luca Grillotti, Rujikorn Charakorn, Sebastian Risi, David Ha, Robert Tjarko Lange

Sakana AI, Tokyo, Japan
yutaro.yamada.y@gmail.com, robert@sakana.ai

Abstract

Biological neural circuits obey Dale's principle: each neuron's synapses are uniformly excitatory or inhibitory. Artificial networks that respect this constraint must coordinate separate excitatory and inhibitory populations, fundamentally changing how credit is assigned during learning. Several biologically plausible learning rules avoid backpropagation's ~~weight-transport~~ weight transport requirement, but ~~achieving~~ it has been difficult to achieve strong performance under Dale's principle beyond MNIST~~has been difficult~~. Error Diffusion (ED) was originally proposed in a dual-stream excitatory/inhibitory architecture, where learning is driven by routing global error signals to all layers without transporting transposed forward weights or relying on random feedback matrices. Whether such a rule can scale under Dale's principle across both supervised classification and reinforcement learning remains unknown. Here, we introduce modulo error routing to extend Error Diffusion beyond binary classification, and show that a dual-stream excitatory/inhibitory architecture trained with ~~ED~~ this method achieves 96.7% on MNIST and establishes a ~~robust~~61.7% baseline on CIFAR-10, demonstrating that ~~competitive~~ representation learning is possible even when strictly enforcing Dale~~'s~~'s principle. For the classification setting, we introduce three domain-specific innovations: layer-specific sigmoid widths, ~~mean-centered~~ batch-centered class error signals, and asymmetric initialization. ~~An~~, and ablation analysis reveals that their relative importance *reverses* between MNIST and CIFAR-10, exposing task-dependent credit-assignment bottlenecks ~~that are~~ invisible to single-benchmark evaluation. In reinforcement learning, ~~by contrast, we retain the dual-stream architecture and multidimensional error routing but relax the strict non-negativity constraint, in this regime, ED integrated effectively~~ we integrate ED with Proximal Policy Optimization (PPO) ~~,~~and evaluate it on continuous-control tasks in Google Brax and on Craftax, an open-ended exploration task. We show that ED-PPO achieves competitive performance ~~demonstrating that the core architecture generalises across domains even as the optimal auxiliary mechanisms shift~~ relative to Direct Feedback Alignment, a backpropagation-free baseline.

©2026 Yutaro Yamada, Luca Grillotti, Rujikorn Charakorn, Sebastian Risi, David Ha, Robert Tjarko Lange. Published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Introduction

In biological neural circuits, the separation of excitatory and inhibitory neuronal populations is a fundamental organizational principle, formalized as Dale's principle: each neuron releases the same neurotransmitter at all of its synapses (Dale, 1935; Eccles, 1976; Strata and Harvey, 1999). This constraint shapes cortical computation through balanced excitation and inhibition ~~, and its disruption is implicated in neurological disorders~~ (Markram et al., 2004). Artificial neural networks, by contrast, freely assign positive and negative weights to any connection, a simplification that enables efficient credit assignment via backpropagation but bears little resemblance to biological learning.

Training deep networks with backpropagation requires the backward pass to use exact transposes of forward weight matrices, the *weight transport problem*, which lacks biological support (Crick, 1989; Bengio et al., 2015). Several alternatives address this concern. Feedback Alignment (FA) replaces backward weights with fixed random matrices (Lillicrap et al., 2016), Direct Feedback Alignment (DFA) routes output errors directly to each hidden layer through fixed random matrices (Nøkland, 2016), and predictive coding frameworks approximate backpropagation through local Hebbian updates (Whittington and Bogacz, 2017; Sacramento et al., 2018). While these methods avoid weight transport, none enforce Dale's principle: all allow arbitrary sign weights. Error Diffusion (ED), originally proposed by (Kaneko, 2000), is a local learning rule in which weight updates depend on presynaptic activity, a postsynaptic activation derivative, and a single global error sign (Fujita, 2026). This locality makes ED naturally compatible with biological constraints, but prior work demonstrated ~~it~~ its effectiveness only on binary classification ~~,~~and a simple classification task like MNIST (Fujita, 2026).

The central question we address is whether a biologically plausible architecture that enforces Dale's principle can achieve competitive performance across both supervised classification and reinforcement learning, and what the task-dependent importance of its components reveals about credit assignment under biological constraints. We make three contributions:

1. A dual-stream excitatory/inhibitory ED architecture that maintains non-negative weights across classification (MNIST, CIFAR-10) and reinforcement learning (Brax locomotion tasks, Craftax) by extending the original ED via modulo error routing.
2. For the classification setting, three domain-specific innovations: layer-specific sigmoid widths, ~~mean-centered~~ batch-centered class error, and asymmetric initialization, whose cross-task ablation reveals that credit-assignment bottlenecks shift qualitatively with task difficulty.
3. ~~A cross-domain evaluation showing~~ Cross-domain evaluation shows that ED-PPO achieves ~~comparable returns~~ returns comparable to backpropagation-based ~~PPO on~~ and DFA-based PPO on continuous control tasks such as Ant, Humanoid~~and HalfCheetah, while DFA fails on the most complex we task (Craftax, $p = 0.010$ vs , and HalfCheetah. On the more complex Craftax task, ED-PPO ~~), demonstrating that the structured error routing of ED generalizes more robustly than random feedback pathways.~~ achieves performance comparable to DFA.

Related Work

Biologically plausible learning rules. The weight transport problem has motivated a family of alternatives to backpropagation (Crick, 1989; Bengio et al., 2015). Feedback Alignment uses fixed random backward weights (Lillicrap et al., 2016), and Liao et al. (2016) showed that approximate weight symmetry can support learning in deeper networks. DFA projects output errors directly to hidden layers (Nøkland, 2016), and Launay et al. (2020) demonstrated that DFA scales to modern architectures, including transformers and convnets. Difference target propagation computes layer-wise learning targets instead of gradients (Lee et al., 2015). Predictive coding provides a complementary framework through local Hebbian updates (Whittington and Bogacz, 2017), and dendritic cortical microcircuit models approximate backpropagation with segregated dendrites (Sacramento et al., 2018). Bartunov et al. (2018) found that biologically motivated algorithms generally struggle to scale beyond simple benchmarks, and Richards et al. (2019) argued that biological plausibility and computational performance need not be in tension if the right architectural principles are identified. Filipovich et al. (2022) demonstrated DFA on photonic hardware, showing that bioplausible rules can enable novel computing substrates.

Dale's principle in neural networks. Song et al. (2016) introduced a framework for training excitatory-inhibitory recurrent networks with Dale's principle, ~~re-parameterizing weights as $W = |W'| \odot M$ where~~ separating synaptic magnitudes from signs by parameterizing weights as $W = W^{+} \odot M$, where $W^{+} \geq 0$ contains trainable connection magnitudes and M is a fixed sign mask ~~,~~ encoding each unit's excitatory or inhibitory identity. Cornford et al. (2021) proposed Dale's ANNs, feedforward architectures with separate excitatory and inhibitory populations that can learn comparably to standard ANNs by treating inhibition as a normalization mechanism. Grossberg (1987) explored competitive learning in networks with separate excitatory and inhibitory interactions. ~~Our approach differs in using separate excitatory and inhibitory populations (dual-stream) rather than a sign mask, which allows independent representations in each stream. Error Diffusion (Dellaferrera and Kreiman, 2022) avoids~~ Error Diffusion (Fujita, 2026; Kaneko, 2000) is also naturally Dalean, representing positive and negative components through separate non-negative populations while avoiding both weight transport and random feedback~~, motivating our~~. Building on this framework, our work extends Error Diffusion to multi-class ~~and RL extensions.~~ classification and reinforcement learning.

Reinforcement learning without backpropagation. Evolution Strategies (ES) provide a gradient-free alternative to policy gradient methods (Salimans et al., 2017), but scale poorly with parameter count. Proximal Policy Optimization (PPO) is a widely used policy gradient method (Schulman et al., 2017) that typically relies on backpropagation for gradient computation. Biologically plausible alternatives for RL credit assignment have been proposed—reward-modulated Hebbian rules can solve decision-making tasks by gating synaptic updates with a global reward signal (Pfeiffer et al., 2010), and Izhikevich (2007) showed that linking spike-timing-dependent plasticity with dopamine signaling addresses the distal reward problem—but these methods have not been demonstrated with deep networks on continuous-control benchmarks. We replace backpropagation in PPO's gradient computation with Error Diffusion, evaluating on Brax physics tasks (Freeman et al., 2021) and Craftax (Matthews et al., 2024).

Method

Dual-Stream Architecture

We adopt the original Error Diffusion formulation proposed by (Kaneko, 2000). To enforce Dale's principle~~(all weights being non-negative)~~, we split each layer into an excitatory ($\mathbf{p}$) and inhibitory ($\mathbf{n}$) stream. The forward pass for layer i computes:

$$\mathbf{p}_i = \phi_i(+\mathbf{p}_{i-1}W_{pp} - \mathbf{n}_{i-1}W_{np} + \mathbf{b}_p)$$
$$\mathbf{n}_i = \phi_i(+\mathbf{n}_{i-1}W_{nn} - \mathbf{p}_{i-1}W_{pn} + \mathbf{b}_n)$$

where all weight matrices $W_{pp}, W_{np}, W_{nn}, W_{pn} \geq 0$ element-wise, ~~and~~ bias parameters $\mathbf{b}_p$ and $\mathbf{b}_n$ are not

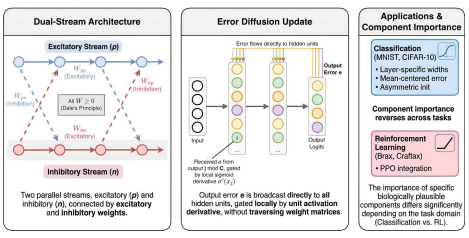


Figure 1: Overview of the dual-stream Error Diffusion framework. **Left:** The excitatory/inhibitory architecture maintains separate streams ($\mathbf{p}$, $\mathbf{n}$) with four non-negative weight matrices per layer, enforcing Dale's principle structurally. **Center:** The Error Diffusion update broadcasts output error directly to hidden layers without weight transport or random feedback matrices. **Right:** The shared architecture is applied to classification (with domain-specific sigmoid widths, ~~mean-centered~~ batch-centered class error, and asymmetric initialization) and reinforcement learning (with ~~ReLU activations and~~ PPO integration).

necessary non-negative. ϕ_i is a layer-specific activation function. The negation signs in front of W_{np} and W_{pn} are structural (hardcoded), ensuring that cross-stream connections are inhibitory while all learnable parameters remain non-negative. This dual-stream design requires four weight sub-matrices per layer, resulting in ~4× more parameters than an unconstrained single-stream network of the same width (e.g., ~32M vs ~8M for DFA on the same architecture).

Error Diffusion Update Rule

The ~~weight update follows the ED principle: each weight is adjusted proportionally to the presynaptic activity, the postsynaptic activation derivative, and a global error signal:~~

$$\Delta W \propto \eta \cdot \mathbf{n}_{\text{pre}}^{\top} \cdot \phi'(z_{\text{post}}) \cdot e_j$$

~~where ϕ' is the activation derivative~~ED update replaces backpropagated layerwise errors with an output-space error signal that is routed directly to hidden units. While the original formulation ~~is~~ was developed for binary classification (Kaneko, 2000; Fujita, 2026), we extend ED ~~from the binary to the~~ to multi-output ~~dimension in the following way. The error signal is computed from the output layer: each hidden unit j receives the error from output unit $j \bmod C$ (where prediction by assigning each hidden unit a fixed output channel. For a hidden unit i, we define $r(i) = i \bmod C$,~~ where C is the output dimension, and use the routed error component $s_{r(i)}$ as its learning signal. (For other design choices of this error routing and why, see Figure 4)

For a mini-batch, let $S \in \mathbb{R}^{B \times C}$ denote the output error signal, and let $M \in \{0,1\}^{H \times C}$ be the fixed routing matrix with $M_{ic} = 1$ iff $r(i) = c$. The routed hidden error is

$$R = SM^{\top}, \tag{1}$$

and the corresponding matrix of local postsynaptic drives is

$$U_p = \phi'(Z_p) \odot R, \tag{2}$$

where $Z_p \in \mathbb{R}^{B \times H}$ is the positive-stream preactivation matrix.

Let $A_p \in \mathbb{R}^{B \times K}$ denote the positive-stream presynaptic activations feeding the layer. In fully connected notation, the positive-to-positive ED update is

$$\Delta W_{pp} \propto A_p^{\top} U_p. \tag{3}$$

In the supervised image classifiers, $C = 10$ and S is the batch-centered one-vs-all classification error. In Craftax policy networks, C is the number of ~~output units), providing coarse~~ discrete action logits. In MuJoCo policy networks, C is the policy distribution parameter dimension for the TanhNormal policy.

Each local weight update is proportional to presynaptic activity, and this routed postsynaptic drive, with stream-specific signs for the dual positive/negative pathways. This deterministic modulo routing provides coarse output-associated credit assignment without requiring per-synapse error gradients, weight transport, or transporting forward weights or using random feedback matrices. ~~This modular routing is deterministic and fixed, in contrast to DFA, which uses random projection matrices that are also fixed but lack structured correspondence to the output.~~ Unlike DFA, whose fixed feedback matrices are random, ED uses a structured correspondence between hidden units and output dimensions.

Classification-Specific Innovations

For the classification setting (MNIST, CIFAR-10), we introduce three domain-specific innovations that address failure modes specific to multi-class classification under Dale's principle. These three innovations are specific to the classification setting and are not used in the RL extension.

Layer-specific sigmoid widths. In the classification architecture, ~~$\phi_i(z) = 1/(1 + e^{-z/\alpha_i})$, with layer-specific~~ $\phi_i(z) = 1/(1 + e^{-2z/\alpha_i})$ with layer-type-specific sigmoid width α_i. The vanishing gradient problem in sigmoid networks is well documented (Roodschild et al., 2020; Ding et al., 2018), and various parameterized activations have been proposed (Szandała, 2020; Hammad, 2024). Since the sigmoid derivative directly gates the error signal in ED (Eq. 3), gradient attenuation is especially severe: post-hoc analysis reveals a 25× decay from output to first hidden layer. Wider sigmoids (larger α_i) maintain larger derivatives, preventing premature saturation. We set $\alpha = 3.0$ for CIFAR-10 convolutional layers and $\alpha = 6.0$ for fully connected layers, including the output layer; MNIST uses only the $\alpha = 6.0$ fully connected setting.

~~Mean-centered~~ **Batch-centered class error signals.** ~~In 10-class classification with one-hot targets, the raw error $e = y - \hat{y}$ contains one positive and nine negative entries per sample. We subtract~~ For the 10-way classification tasks, the ED update uses independent sigmoid output activations rather than a softmax. For a mini-batch of size B, we form a one-vs-all output error $E_{bc} = \mathbf{1}[y_b = c] - \hat{y}_{bc}$, where $\hat{y}_{bc}$ is the positive-stream output activation for class c. Since each example contributes one target class and nine non-target classes, the raw one-vs-all error can contain a strong class-wise offset, especially early in training. We therefore subtract the mini-batch mean separately for each class: $\tilde{E}_{bc} = E_{bc} - \frac{1}{B}\sum_{i=1}^{B} E_{ic}$. This makes the error signal zero-mean across the batch for every class, reducing persistent suppression or excitation of class channels caused by the ~~per-class mean : $e_{\text{centered}} = (y - \hat{y}) - \text{mean}(y - \hat{y})$.~~ This rebalances the positive and negative error components, preventing the update from being dominated by suppressing incorrect classes, one-vs-all target imbalance. The centered signal is then applied directly to the output layer, with the negative stream receiving the opposite sign; hidden fully connected units and convolutional channels receive class-routed copies of the same centered error.

Asymmetric E/I initialization. For ~~feedforward~~ convolutional and fully connected layers, weight initialization samples weight parameters uniformly between ~~0~~ parameters are initialized from a non-negative uniform distribution in $[0, 1)$ and ~~$(-1, 0]$~~. Afterwards, these parameters are additionally ~~$\times$~~scaled by the ~~number of incoming units. Finally,~~ inverse square root of the fan-in. Hidden excitatory weights (W_{pp}, W_{nn}) are ~~scaled~~ then scaled by $1.5\times$~~and~~ , while inhibitory weights (W_{np}, W_{pn}) are scaled by $0.5\times$, ~~creating an initial~~ creating an expected E/I ~~ratio of approximately 3:1~~. ~~The~~ scale ratio of $3:1$. The final fully connected output layer uses symmetric initialization ($1{+}1.0\times$ for both excitatory and inhibitory weights) to avoid output saturation.

~~Figure 1 summarizes the error diffusion training procedure and methodological contributions.~~

RL Extension: ED-PPO

For reinforcement learning, we integrate ~~the~~ a Dale-constrained dual-stream ~~ED~~ Error Diffusion (ED) architecture into PPO~~(Schulman et al., 2017)~~. The policy and value networks ~~both use the~~ are both dual-stream architecture with ReLU activations and RMS stream normalization between layers. Input observations are split into MLPs with separate positive and negative pathways and non-negative synaptic weights. Each layer computes excitatory-minus-inhibitory preactivations for the positive and negative streams~~via $\mathbf{p}_i = \text{ReLU}(z_i^+)$, $\mathbf{n}_i = \text{ReLU}(-z_i^-)$, and the output is computed as $\hat{y} = \mathbf{p}_L - \mathbf{n}_L$~~. ED provides the gradient signal in place of backpropagation : for the policy network (vector output), the same modular routing applies, applies a width-scaled sigmoid nonlinearity, and combines the final streams as $\hat{y} = y^{+} - y^{-}$. Both Dale implementations initialize the two streams symmetrically from the same observation, $\mathbf{p}_0 = \mathbf{n}_0 = x$, after optional stream normalization, rather than splitting inputs with ReLU.

ED replaces backpropagation through hidden layers after the PPO's objective function supplies an output-level error signal. For vector-valued policy outputs, the error is routed to hidden units by output class/channel assignment; for the value network (scalar output)scalar value network, the error ~~broadcasts directly~~ is broadcast to all hidden units. ~~Weights.~~All weight matrices are initialized non-negative ~~via $|U| \sim |\mathcal{N}(0, \sigma^2)|$ with asymmetric scaling (excitatory at $1.0\times$, inhibitory at $0.1\times$)~~using absolute Gaussian samples,

with separate scaling for excitatory and inhibitory pathways, and weights are clamped non-negative after optimizer updates.

Figure 1 summarizes the error diffusion training procedure and methodological contributions.

Experimental Protocol

Classification. We evaluate six variants on MNIST and CIFAR-10: (1) the **proposed ED** with all three innovations, (2) a **DFA baseline** using fixed random feedback matrices without Dale's constraints, (3) the **seed ED** (the dual-stream architecture without any of the three innovations, i.e., using uniform sigmoid width $\alpha = 1.0$, raw error signals, and symmetric initialization), and three ablations removing one innovation each: (4) **no ~~mean-centered~~ batch-centered class error**, (5) **symmetric init**, and (6) **uniform sigmoid width**. Each variant is trained for 250 epochs with five random seeds per task (60 runs total). For classification, weights are clamped to a floor of 10^{-4} after each update to enforce Dale's principle.

Reinforcement learning. We compare ED-PPO against BP-PPO (standard backpropagation~~and ES~~, DFA-PPO (Direct Feedback Alignment) (Nøkland, 2016), and ES (Salimans et al., 2017) on three Brax (Freeman et al., 2021) locomotion tasks (Ant, HalfCheetah, Humanoid)~~and Craftax (Matthews et al., 2024)~~. On Craftax~~, we additionally compare against~~ (Matthews et al., 2024), we compare against BP-PPO and DFA-PPO. Additionally, we evaluate a non-Dalean variant of ED-PPO, in which weights are not constrained to be non-negative, and denote it as "ED-PPO (non-Dalean)". Each algorithm is evaluated with five random seeds per environment. We report the final episode reward (the reward at the last evaluation checkpoint) averaged over seeds; statistical comparisons use Welch's t-test with $\alpha = 0.05$. The RL experiments use a separate training and evaluation pipeline from the classification experiments, with both pipelines sharing the same dual-stream ED architecture but differing in activation functions, normalization, and optimization procedure as described above.

Results

Classification: Main Comparison

Figure 2 summarizes accuracy across all six classification variants. The proposed ED achieves $96.7 \pm 0.1\%$ on MNIST and $61.7 \pm 0.7\%$ on CIFAR-10, a substantial improvement over seed ED ($50.4 \pm 9.8\%$ and $11.6 \pm 2.2\%$). DFA achieves higher accuracy on both tasks (97.6% and 69.1%) but violates Dale's principle, requiring $\sim$2.84M negative weights. The accuracy gap between ED and DFA widens from 0.9 pp on MNIST to 7.4 pp on CIFAR-10, suggesting that the cost of maintaining non-negative weights grows with task difficulty. We note that ~~62%~~ $\sim$62% test accuracy on CIFAR-10 is still far from competitive compared to traditional gradient-based methods. That being said, this is the first time ED has been successfully applied to convolutional neural networks. Previously, Fujita (2026) obtained ~~$\sim$55.2%~~ $\sim$55.2% test accuracy on CIFAR-10 using a purely feedforward MLP with flattening.

Table 1: Ablation effect sizes (Δ = proposed ED minus ablation, in percentage). Negative values indicate accuracy loss.

Removed Component	MNIST Δ	CIFAR-10 Δ
~~Mean-Centered~~ Batch-Centered Error	-0.3	-47.9
Layer-Specific Widths	-71.4	-15.1
Asymmetric Init	$+0.0$	-5.5

Classification: Cross-Task Ablation Reversal

In the classification setting, ablation analysis reveals that the importance hierarchy of the three innovations reverses between tasks (Figure 2, Table 1).

MNIST. Removing layer-specific sigmoid widths is catastrophic (96.7% $\rightarrow$ 25.3%, $\Delta = -71.4$ pp), collapsing accuracy to near-chance. Removing ~~mean-centered~~ batch-centered class error causes only $\Delta = -0.3$ pp, and symmetric initialization has no measurable effect ($\Delta = +0.0$ pp). Gradient flow regulation is the sole bottleneck on this easier task.

CIFAR-10. The hierarchy reverses. Removing ~~mean-centered~~ batch-centered error is now the most destructive ablation (61.7% $\rightarrow$ 13.8%, $\Delta = -47.9$ pp), causing four of five seeds to collapse. Uniform sigmoid width causes $\Delta = -15.1$ pp, and symmetric initialization causes $\Delta = -5.5$ pp. All three innovations contribute, but their relative ordering changes fundamentally.

Interpretation. This reversal reflects qualitatively different credit-assignment bottlenecks. MNIST's well-separated features allow learning even without error centering~~,~~ but without wide sigmoids, the derivatives vanish entirely. On CIFAR-10, higher inter-class similarity makes the 9:1 error imbalance overwhelming: ~~mean-centering~~ batch-centering becomes essential to prevent uniform output suppression. The reversal demonstrates that evaluating biologically plausible methods on a single benchmark may obscure critical ~~design~~ trade-offs.

~~Reinforcement Learning~~RL: Cross-Domain Generalization

Figures 3 and 5 ~~present ED-PPO~~ report performance across four RL ~~environments~~tasks. ED-PPO ~~performs~~ is strongest on HalfCheetah, where it outperforms BP-PPO (5494 ± 691 vs 3520 ± 485; $p < 0.001$), ED-PPO (non-Dalean) ($p = 0.028$), and ES ($p = 0.003$), while matching DFA-PPO (5581 ± 359). On Ant it is on par with both PPO variants (6891 ± 835), significantly exceeding only ES ($p = 0.004$). On Humanoid (6670 ± 2592) and Craftax (20.9 ± 2.9) it trails BP-PPO and ED-PPO (non-Dalean), though the Craftax gap to the latter is not significant.

ED-PPO (non-Dalean) also performs competitively across the same four environments, on par with BP-PPO on Humanoid (9804 ± 1144 vs 8478 ± 3252). On Ant, ED-PPO (non-Dalean) achieves a higher mean (7616 ± 2031 vs 6740 ± 1781) but the difference is not ~~statistically significant ($p = 0.244$)~~ significant due to high variance. On HalfCheetah, ED-PPO (non-Dalean) achieves a similar score (3498 ± 1703 vs 3520 ± 485)~~, with one of five seeds collapsing to low reward. On Craftax, BP-PPO significantly leads (27.0 ± 0.9 vs 20.9 ± 1.5; $p = 0.002$). ES consistently ranks below both PPO variants on all Brax tasks~~. Notably, DFA-PPO on Craftax achieves only 19.8 ± 1.5, significantly below ED-PPO (non-Dalean) ($p = 0.010$), demonstrating that random feedback pathways that suffice for simpler tasks fail on complex, open-ended environments.

ED-PPO ~~exhibits substantially~~ (non-Dalean) exhibits higher variance than BP-PPO ~~across all environments (e.g., Ant std: 2031 vs 835~~on Ant and HalfCheetah, though on Humanoid its variance is lower (1144 vs 3252), suggesting that the ED gradient signal ~~introduces~~ can introduce additional stochasticity. ~~On HalfCheetah, one ED-PPO seed collapses to low reward (943), inflating variance and reducing the mean below BP-PPO.~~ The Craftax shortfall (-4.0, $p = 0.002$) and higher variance across environments suggest that the coarse modular error routing of ED may be less reliable for tasks requiring fine-grained temporal credit assignment or stable convergence.

DFA Across Domains

DFA provides an informative comparison point across both classification and RL. In classification, DFA achieves the highest accuracy (97.6% MNIST, 69.1% CIFAR-10) but violates Dale's principle. On Craftax, DFA-PPO is the weakest method (19.8 vs BP-PPO 27.0, ED-PPO 23.0), demonstrating that random feedback pathways that work for supervised learning can fail for complex RL. This cross-domain pattern—DFA strong on classification, weak on complex RL—parallels the finding that method components assume different importance under different task demands.

~~Mechanistic Evidence~~Post-hoc Analysis

~~Gradient~~ Surrogate gradient attenuation. Post-hoc analysis of training dynamics on CIFAR-10 (Figure 6a) reveals local surrogate gradient magnitudes dropping from 6.6×10^{-3} at the output layer to 2.7×10^{-4} at the first hidden layer—a $25\times$ attenuation. This pattern is visible from epoch 3, indicating a structural property of sigmoid-gated error diffusion that motivates layer-specific widths. Per-layer learning rate multipliers ($3.0\times$ early, $0.5\times$ output) partially compensate but cannot substitute for wider sigmoid derivatives. The near-zero generalization gap (0.98%) despite $\sim$32M nominal parameters is consistent with underfitting rather than overfitting, but the cause is ambiguous: the 37.3% of weights at floor (10^{-4}) reduces effective capacity well below the nominal parameter count, so the limitation may reflect constrained capacity rather than (or in addition to) credit assignment quality.

Emergent E/I balance. Weight-level E/I ratios (Figure 6b) reveal that training drives the 3:1 asymmetric initialization toward near-balanced ratios ($\sim$1.0) in hidden layers. A depth-dependent gradient emerges: the first layer reaches near-perfect balance (E/I = 1.03), the second becomes slightly inhibitory-dominant (0.90), and the third develops the strongest inhibitory bias (0.81). This increasing inhibition with depth is loosely ~~analogous to biological observations of inhibitory interneuron density increasing in higher cortical areas~~ consistent with biological observations that inhibitory circuitry and E/I balance vary systematically across cortical hierarchies (Markram et al., 2004), though weight-level E/I ratios are an indirect proxy for biological E/I balance, which involves cell counts, firing rates, and synaptic strengths jointly. The convergence toward balance explains the task-dependent importance of asymmetric initialization: on MNIST, balance is reached quickly regardless of initial conditions ($\Delta = +0.0$ pp); on CIFAR-10, the asymmetric head start prevents early instability ($\Delta = -5.5$ pp).

Implicit sparsity. The non-negative weight floor induces substantial implicit sparsity: 37.3% of weights reach the floor (10^{-4}) after training. Inhibitory (cross-stream) FC connections are pruned most aggressively (up to 68.8% at the floor), compared to 26–49% for excitatory connections. Convolutional layers are much less affected (<1% to 18%). This asymmetric pruning suggests that ~~Dale's principle~~ the non-negative Dale-style parameterization introduces a structured capacity bottleneck that preferentially ~~eliminates~~ suppresses inhibitory connections in deeper layers.

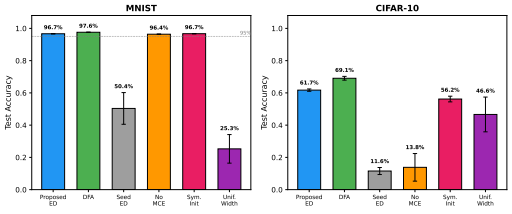
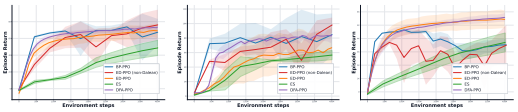


Figure 2: Classification accuracy across all six variants on MNIST (left) and CIFAR-10 (right). Error bars: ± 1 std over 5 seeds. The ablation importance hierarchy reverses between tasks: layer-specific widths dominate on MNIST, ~~mean-centered~~ batch-centered class error dominates on CIFAR-10.



(a) Ant (b) Humanoid (c) HalfCheetah

Figure 3: ~~Final reward~~ Episode return across three RL environments~~comparing ED-PPO, BP-PPO, and ES~~. Error bars indicate ± 1 standard deviation over 5 seeds. ~~The comparison between ED-PPO and BP-PPO on Humanoid is not statistically significant ($p = 0.42$), differences on Ant and HalfCheetah are not statistically significant.~~

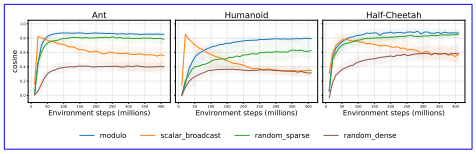


Figure 4: Cosine similarity between the ED local update direction and the BP gradient direction, restricted to policy trunk parameters and excluding the policy in a dual-stream architecture, over PPO training on Ant, Humanoid, and Half-Cheetah. Lines show the mean over 5 seeds; shaded regions show the seed-wise min-max range. `modulo` uses deterministic one-output-per-hidden-unit routing, `random_sparse` uses random one-output-per-hidden-unit routing, `scalar_broadcast` sends the same normalized sum of output errors to every hidden unit, and `random_dense` denotes the normalized dense random feedback variant used in (Launay et al., 2019). Higher cosine indicates closer agreement between ED and BP trunk parameter-gradient directions.

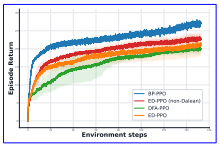


Figure 5: Episode return on Craftax comparing ED-PPO, BP-PPO, and DFA. Error bars: ± 1 std over 5 seeds.

ers.

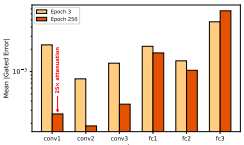
Discussion

Summary. The central finding of this work is that the dual-stream ED architecture is viable across both classification and reinforcement learning, but the auxiliary mechanisms required for competitive performance are domain-dependent. In classification, ~~mean-centered~~ batch-centered class error and layer-specific sigmoid widths are critical stabilizers whose relative importance reverses between MNIST and CIFAR-10. In RL, the same dual-stream architecture with ReLU activations and RMS normalization, without sigmoid widths or ~~mean-centering~~batch-centering, achieves competitive performance on locomotion tasks, showing comparable but variable rewards on Ant, Humanoid, and HalfCheetah.

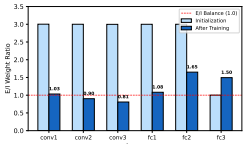
Implications for biological plausibility research. Single-benchmark evaluations can produce misleading conclusions about which architectural features matter. The ablation reversal between MNIST and CIFAR-10 demonstrates that a component critical for one task can be negligible for another, and the DFA cross-domain pattern (strong on classification, weak on complex RL) reinforces this point. The emergent E/I balance convergence, from 3:1 toward $\sim$1:1 with a depth-dependent inhibitory gradient, provides a connection to biological self-organization. Cortical circuits are known to develop balanced E/I ratios during maturation through intrinsic and synaptic homeostatic mechanisms (Markram et al., 2004; Turrigiano, 2011), and our results suggest that a similar homeostatic process can emerge from gradient-driven learning under Dale's principle constraints, without explicit balance-enforcing mechanisms. The success of ED-PPO on locomotion tasks despite using ReLU rather than sigmoid activations raises an interesting question about the relationship between activation function choice and credit assignment quality. In classification, the sigmoid derivative explicitly gates the error signal, making the width parameter critical. In RL, ReLU's unbounded positive derivative may provide a natural gradient pathway that avoids the attenuation problem without requiring layer-specific tuning. This activation-dependent difference in error flow may partially explain why the classification-specific innovations are unnecessary in the RL setting.

Limitations. The classification innovations (sigmoid widths, MCE, asymmetric init) are specific to the classification setting; we do not claim they transfer to RL. ED-PPO shows substantially higher variance than BP-PPO across all environments, suggesting that the coarse error routing introduces stochasticity that may limit reliability. The Craftax shortfall (-4.0 reward vs BP-PPO) indicates that ED's modular credit assignment may be insufficient for tasks requiring fine-grained temporal reasoning. The accuracy gap between ED and DFA on classification (-0.9 to -7.4 pp) quantifies the current cost of enforcing Dale's principle. On Craftax, DFA underperforms ED-PPO, suggesting that the cost of Dale's principle is task-dependent. Finally, the dual-stream architecture's $\sim 4\times$ parameter overhead may itself limit effective capacity, particularly on classification where 37.3% of weights are pruned to the floor. From the perspective of adaptive computation, the emergent behaviors we observe—E/I balance self-organization, asymmetric pathway pruning, and task-dependent component criticality—suggest that biologically constrained architectures may develop internal regulatory mechanisms analogous to those found in living systems.

Future Work. Several directions follow naturally from the results presented here. First, ~~the non-negativity constraint imposed by~~ Dale's principle may ~~translate into concrete hardware advantages: 1-sign-free arithmetic simplifies multiply-accumulate circuitry and could enable more efficient analog~~ offer hardware advantages by constraining synaptic weights themselves to be non-negative. In our Error Diffusion formulation, negative contributions are still required, but their sign is determined by fixed excitatory/inhibitory pathway structure rather than by arbitrary signed weights. Thus, the benefit is not the elimination of subtraction or differential circuitry altogether, but the replacement of unconstrained signed synapses with non-negative synaptic magnitudes and fixed-sign routing. This may be especially relevant for analog, photonic, or synapse-device-based neuromorphic substrates, ~~complementing~~ where physical synaptic elements often naturally encode non-negative quantities, while sign can be implemented at the level of population identity, optical/electrical phase, or excitatory/inhibitory summation. This complements prior demonstrations of bioplausible rules on photonic hardware (Filipovich et al., 2022). Second, the implicit sparsity we observe



(a) Per-layer ~~gradient~~ local surrogate-gradient magnitudes at epochs 3 and 250 on CIFAR-10 (log scale). The 25× attenuation from output to first hidden layer motivates layer-specific sigmoid widths.



(b) E/I weight ratio at init (3:1 for hidden layers) vs. after 250 epochs. Hidden layers converge toward balance; deeper layers become inhibitory-dominant.

Figure 6: Post-hoc ~~mechanistic~~ analyses of the proposed ED model on CIFAR-10. (a) ~~Gradient~~ Local surrogate gradient attenuation across layers. (b) Emergent E/I balance from asymmetric initialization toward biological-like equilibrium.

suggests that Dale-compliant training may yield model compression ~~"~~for free,~~"~~ since weights bounded below at zero cannot recover once suppressed, turning the floor into a natural pruning mechanism that could be exploited with structured sparsity kernels at inference time. Third, we conjecture that the dual-stream architecture may be unusually well-suited to continual and open-ended learning settings, where catastrophic forgetting is a central obstacle: the dedicated inhibitory stream provides a structural mechanism for dampening large gradient excursions, and the sign constraint prevents the kind of unconstrained weight sign flips that are thought to contribute to representational overwriting. Evaluating ED on sequential task streams and lifelong RL benchmarks is a natural next step. Fourth, the segregation of computation into excitatory ~~"amplifiers" and inhibitory "suppressors"~~ "amplifiers" and inhibitory "suppressors" offers a potential interpretability handle that standard networks lack: when a model makes an error, one can ~~,~~in principle trace whether the mistake originated from a feature being falsely amplified by the excitatory stream or insufficiently suppressed by the inhibitory stream, providing a more mechanistic form of attribution than gradient-based saliency on sign-unconstrained networks. Finally, closing the accuracy gap to DFA and BP-PPO on the hardest tasks, whether through richer error-routing schemes, learned (rather than fixed modular) output-to-hidden projections, or hybrid activation functions that preserve sigmoid locality while mitigating attenuation, remains the most direct path toward making Dale-compliant learning competitive with unconstrained backpropagation.

Conclusion

We demonstrated that a dual-stream excitatory/inhibitory architecture trained with Error Diffusion achieves competitive performance across both supervised classification and reinforcement learning while maintaining non-negative weights consistent with Dale's principle. In the classification setting, three domain-specific innovations enable scaling from binary to 10-class problems, and their cross-task ablation reversal reveals that credit-assignment bottlenecks shift qualitatively with task difficulty. In reinforcement learning, the same core architecture integrated with PPO achieves comparable rewards to backpropagation-based PPO on locomotion tasks, without requiring the classification-specific stabilizers, though with substantially higher variance. The cross-domain pattern, competitive performance with domain-dependent auxiliary mechanisms, suggests that biologically plausible learning rules need not be monolithic: a shared architectural core can support diverse task demands when augmented with appropriate domain-specific components. The accuracy gap relative to ~~DFA and~~ BP-PPO on the hardest tasks quantifies the current cost of Dale's principle compliance and provides concrete benchmarks for future work on narrowing this gap while preserving biological fidelity.

Acknowledgement

Generative AI tools were used in the preparation of the manuscript, including text editing, code generation, image generation, and data analysis assistance. Additionally, we leveraged AI to generate ideas, run autonomous experimentation, and refine hypotheses. All claims, code implementation, manuscript writing, and figures were either reviewed by or created by the authors.

References

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Cornford, J., Kalajdzievski, D., Leite, M., Lamarquette, A., Kullmann, D. M., and Richards, B. A. (2021). Learning to live with dale's principle: {ANN}s with separate excitatory and inhibitory units. In *International Conference on Learning Representations*.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Dale, H. (1935). Pharmacology and nerve-endings. *Proceedings of the Royal Society of Medicine*.

Dellaferrera, G. and Kreiman, G. (2022). Error-driven input modulation: solving the credit assignment problem without a backward pass. In *International Conference on Machine Learning*, pages 4937–4955. PMLR.

Ding, B., Qian, H., and Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841.

Eccles, J. C. (1976). From electrical to chemical transmission in the central nervous system: the closing address of the sir henry dale centennial symposium cambridge, 19 septenber 1975. *Notes and records of the Royal Society of London*, 30(2):219–230.

Filipovich, M. J., Guo, Z., Al-Qadasi, M., Marquez, B. A., Morison, H. D., Sorger, V. J., Prucnal, P. R., Shekhar, S., and Shastri, B. J. (2022). Silicon photonic architecture for training deep neural networks with direct feedback alignment. *Optica*, 9(12):1323–1332.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*.

Fujita, K. (2026). A diagnostic evaluation of neural networks trained with the error diffusion learning algorithm. *Discover Artificial Intelligence*.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63.

Hammad, M. (2024). Deep learning activation functions: Fixed-shape, parametric, adaptive, stochastic, miscellaneous, non-standard, ensemble. *arXiv preprint arXiv:2407.11090*.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452.

Kaneko, I. (2000). Sample programs of error diffusion learning algorithm. `https://web.archive.org/web/20000306212433/http://village.infoweb.ne.jp/~fwhz9346/ed.htm`. [Online; accessed 13-April-2026].

Launay, J., Poli, I., Boniface, F., and Krzakala, F. (2020). Direct feedback alignment scales to modern deep learning tasks and architectures. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Launay, J., Poli, I., and Krzakala, F. (2019). Principled training of neural networks with direct feedback alignment.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. In Appice, A., Rodrigues, P. P., Santos Costa, V., Soares, C., Gama, J., and Jorge, A., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 498–515, Cham. Springer International Publishing.

Liao, Q., Leibo, J., and Poggio, T. (2016). How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276.

Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., and Wu, C. (2004). Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience*, 5(10):793–807.

Matthews, M., Beukman, M., Ellis, B., Samvelyan, M., Jackson, M., Coward, S., and Foerster, J. (2024). Craftax: A lightning-fast benchmark for open-ended reinforcement learning. *arXiv preprint arXiv:2402.16801*.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29.

Pfeiffer, M., Nessler, B., Douglas, R. J., and Maass, W. (2010). Reward-modulated hebbian learning of decision making. *Neural computation*, 22(6):1399–1444.

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., Clopath, C., Costa, R. P., de Berker, A., Ganguli, S., et al. (2019). A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770.

Roodschild, M., Gotay Sardiñas, J., and Will, A. (2020). A new approach for the vanishing gradient problem on sigmoid activation. *Progress in Artificial Intelligence*, 9(4):351–360.

Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

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

Song, H. F., Yang, G. R., and Wang, X.-J. (2016). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology*, 12(2):e1004792.

Strata, P. and Harvey, R. (1999). Dale's principle. *Brain research bulletin*, 50(5-6):349–350.

Szandala, T. (2020). Review and comparison of commonly used activation functions for deep neural networks. In *Bio-inspired neurocomputing*, pages 203–224. Springer.

Turrigiano, G. (2011). Too many cooks? intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement. *Annual review of neuroscience*, 34(1):89–103.

Whittington, J. C. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262.

E. AUTOILLUSTRATE: Supplementary Material

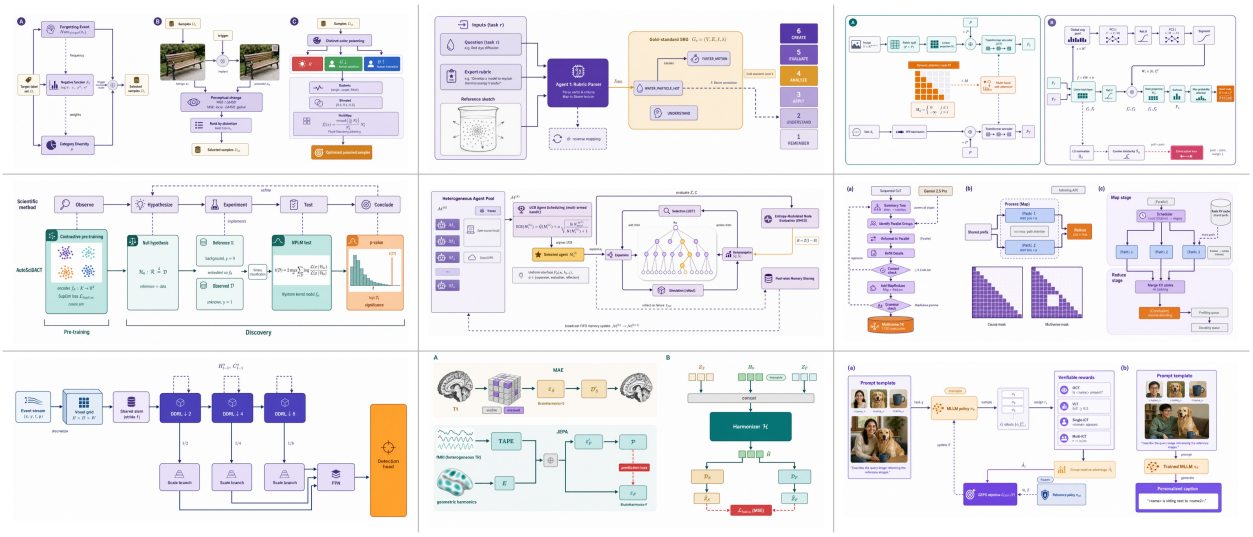


Figure 21 | Examples of figures generated by AUTOILLUSTRATE on PaperBananaBench (Zhu et al., 2026a).

E.1. Hyperparameters

We use a fixed benchmark configuration across the AUTOILLUSTRATE variants, changing only the coding-agent model, critic availability, or effort level indicated by each variant. Table 7 summarizes the shared settings. The original PaperBanana baseline relies on `gemini-3.1-pro-preview` and `gemini-3-pro-image-preview` (Nano Banana Pro). For the PaperBanana "Claude + GPT Image 2" variant, similarly to our pipeline, we use Claude Opus 4.8 for the Stylist, GPT-5.5 for the Critic, and GPT Image 2 for the Visualizer. We retain Gemini for the Retriever because its long inputs exceed Claude's one-million-token context limit, and retain PaperBanana's original Planner because AUTOILLUSTRATE has no corresponding component that accepts images as input.

Table 7 | Hyperparameters for the PaperBananaBench evaluation.

Hyperparameter	**Value**
Codebase version	`autoillustrate` v0.2.0
Coding agent	`Claude Code`; `Codex` for the Codex variant
Agent model	`claude-opus-4-8`; `gpt-5.6-sol` for the Codex variant
Effort level	high
Panel inspects (Phase 3)	3: one inspect + up to two repair passes per panel
Enhancement attempts	4: Phase 5 enhance + 3 verify + 2 repair passes
Audit VLM (Phase 3 inspect, Phase 6 verify Stage A)	`gpt-5.5`
Unit-test backend (Phase 6 Stage B)	`gpt-5.5`
Enhancement model (Phase 5)	`gpt-image-2`, quality `high`
Judge model	`gpt-6-astra`, high reasoning effort (Table 4); `gemini-3.1-pro-preview` (Table 8)

E.2. Results with Gemini-3.1-Pro as VLM judge

Table 8 reports the PaperBananaBench scores of the same generated diagrams when judged by Gemini-3.1-Pro instead of GPT-6-Astra (Table 4). Gemini-3.1-Pro awards higher scores than GPT-6-Astra across all methods. The ablation trends within AUTOILLUSTRATE are preserved, and AUTOILLUSTRATE still outperforms the Gemini-based PaperBanana on Overall score (50.0 vs. 46.7). However, PaperBanana with Claude Opus 4.8 and GPT Image 2 obtains the highest Overall score (56.5) under this judge, mainly due to higher Faithfulness (53.1 vs. 43.7), whereas GPT-6-Astra ranks it last (18.3). This discrepancy stems from PaperBanana's raster diagrams, which frequently contain small image-generation errors, such as disconnected or dangling arrows, that GPT-6-Astra treats as faithfulness vetoes while Gemini-3.1-Pro overlooks them; Figure 22 shows an example diagram together with the Faithfulness rationales of both judges. A likely cause of these errors is that PaperBanana renders complex, highly detailed diagrams with a prompt-only image generation model: the model receives text alone; on the other hand, the image generation model used in AUTOILLUSTRATE takes the rendered editable `draw.io` diagram as an input image.

E.3. Limitations

Table 9 shows representative failure cases of AUTOILLUSTRATE, together with their causes. Another recurring issue is that the enhancement model tends to turn unidirectional arrows into bidirectional ones: for instance, in Figure 20b, the "No weight transport" arrow of panel B acquires a second arrowhead that is absent from the editable diagram (Figure 20a). The enhancement model may also struggle to render uniform background colors for boxes, as can be seen in the same figure.

Table 8 | Mean scores on PaperBananaBench, using Gemini-3.1-Pro as VLM judge.

Method	Variant	Faithfulness	Conciseness	Readability	Aesthetic	Overall	Editable source
AUTOILLUSTRATE	Claude	43.7	45.7	45.0	75.2	50.0	✓
	+ no critic	42.8	41.6	45.2	75.3	46.6	✓
	+ low effort	37.0	43.2	46.2	74.8	42.1	✓
	Codex	42.3	39.9	31.0	49.8	34.9	✓
PaperBanana	Gemini	36.0	**57.0**	48.1	74.7	46.7	✗
	Claude + GPT Image 2	**53.1**	30.0	**51.5**	**79.5**	**56.5**	✗
Human	–	50.0	50.0	50.0	50.0	50.0	✓

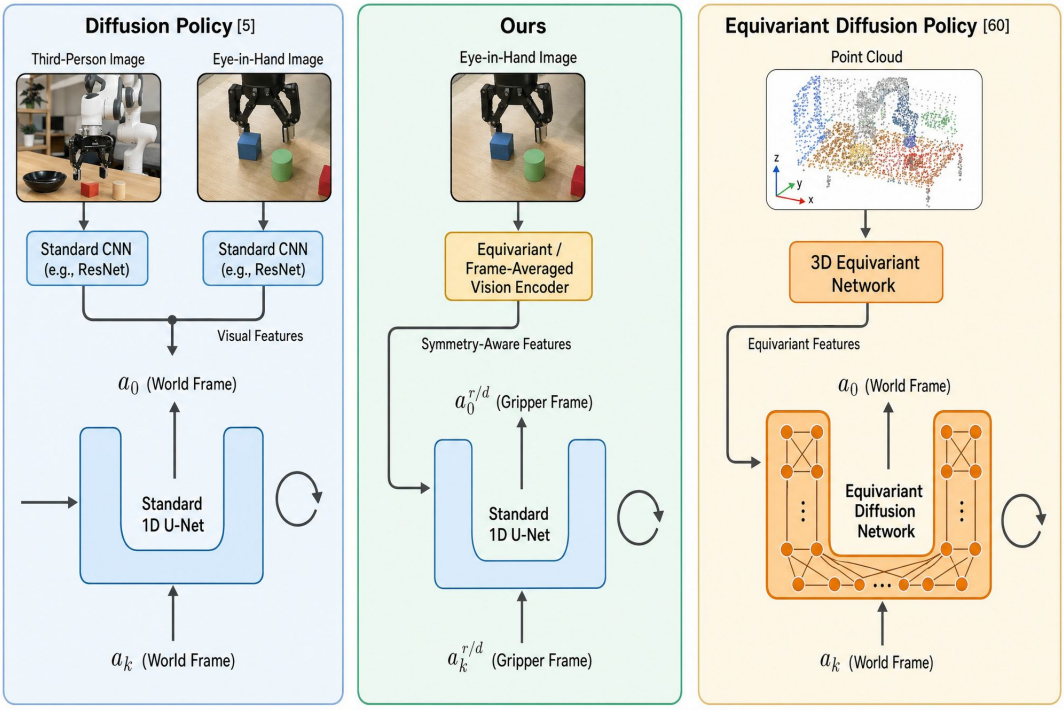
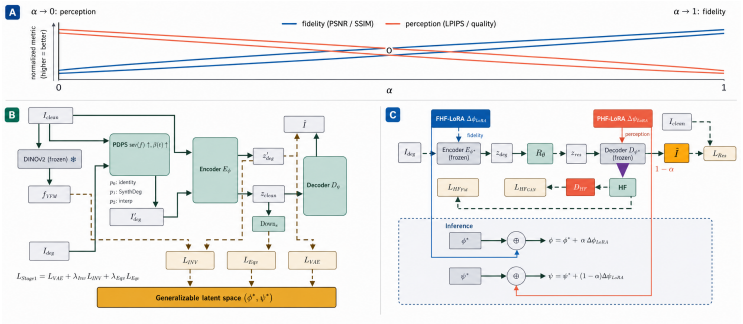
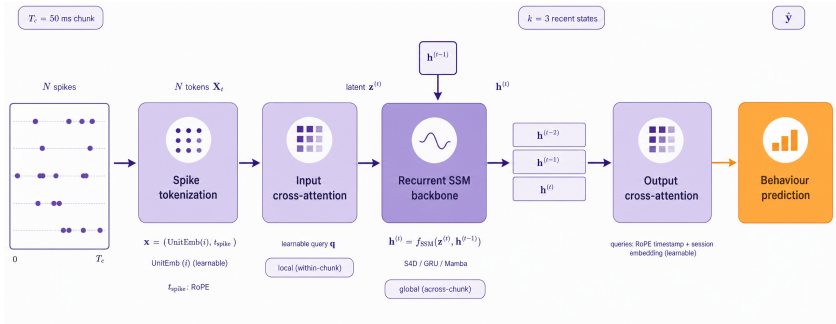
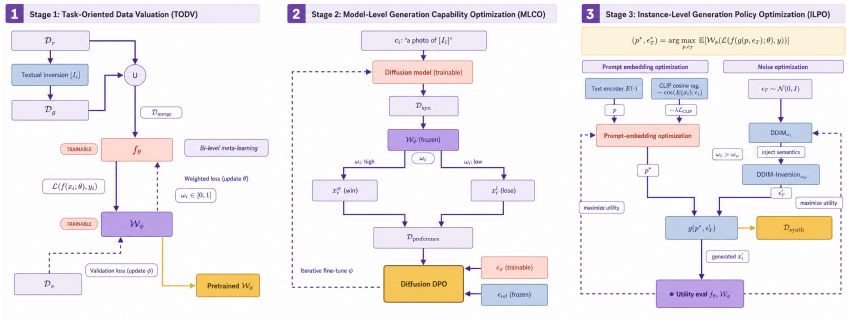


Figure 22 | Example PaperBanana (Claude Opus 4.8 + GPT Image 2) diagram on which the two VLM judges disagree on Faithfulness.
Gemini-3.1-Pro rationale: "Accurately captures the structural differences and integrates specific text details like delta/relative trajectories ($a^{r/d}$) and frame-averaged encoders. It strictly aligns with the caption's visual simplification. No veto errors."
GPT-6-Astra rationale: "Correctly depicts the proposed method's encoder/backbone distinction and action frames. However, in the original Diffusion Policy panel, the visual-feature connection terminates above the clean-action label rather than entering the U-Net. The separate, unlabeled arrow entering the U-Net is disconnected from those features. This omits the essential visual-conditioning connection and constitutes a logical-flow veto, not merely simplification."

Table 9 | Example failure cases of AutoIllustrate.

AutoIllustrate Diagrams	Cause of Failure
	**Edge attached to the wrong source.** The equivariance loss $\mathcal{L}_{Eqv}$ is fed the downsampled latent directly from $Down_s$, bypassing the decoder D_ψ, whereas the method defines the loss on the decoded output $D_\psi(z_{down})$.
	**Space-inefficient layout.** Disconnected annotations float above the main pipeline, yielding a non-rectangular bounding box with large dead zones that waste vertical space under standard publication formatting.
	**Insufficient visual abstraction.** Raw equations are transcribed into the figure rather than distilled, most visibly the full argmax objective placed in a box at the top of Stage 3, and modules are labelled with bare symbols (f_θ, W_ϕ) instead of their function, reducing the diagram to a box-ified restatement of the text.

F. Additional Details on The Agentic Scientist

This section provides additional details about the The Agentic Scientist system, including the prompts used by the ideation, execution, ablation-study, and task-drafting agents.

F.1. Bibliography Retrieval

The paper-writing phase (Section 3.5) never lets the writer agent produce bibliography entries from memory. Before drafting, a reference-planning agent reads the staged run artifacts and writes a small query plan (6–10 search queries by default) covering the central topic, the core method families, the comparator literatures a reviewer would expect, and the benchmark context; this agent does not retrieve anything itself. A deterministic script then executes the plan against the Semantic Scholar Graph API (Kinney et al., 2023): for each query it searches up to 25 candidate papers, ranks them by citation count, fetches the BibTeX entry served by the API, and keeps only candidates with a well-formed entry and citation key, deduplicating by paper identifier and key until a target of 8–15 references (configurable) is met. If Semantic Scholar is unavailable, rate-limited, or returns too few references, the script falls back to OpenAlex (Priem et al., 2022), resolving each work's BibTeX through DOI content negotiation and filling in author names from OpenAlex metadata; Google Scholar via the `scholarly` package is used as a last resort. The script writes a `references.bib` file together with a provenance record listing, for every entry, the originating query, source URL, and retrieval timestamp. The writer agent may only cite keys present in this file, and the validation checks of the repair loop reject any citation key or claim-to-source link that does not resolve to a retrieved entry. After a first draft exists, a gap-analysis agent inspects the manuscript for under-cited passages (e.g., comparator families named in the related work without a citation), a post-draft planner turns these gaps into a few additional queries, the same retrieval script enriches the bibliography incrementally, and the writer revises only the flagged paragraphs.

F.2. Ideation Agent

Ideation Agent Prompt

```
# Role
You are an idea-generation agent for iterative scientific optimization.
Goal: produce one high-quality next idea plus rubric bundle for the execution loop.
Use the `create-idea` and `memory-log` skill workflows for output structure, evidence logging, and checks.

# Task Context
## Goal
{{TASK_GOAL}}

## Rubrics
{{TASK_RUBRICS}}

## Cycle
- cycle_index: `{{CYCLE_INDEX}}`
- global_results_dir: `{{GLOBAL_RESULTS_DIR}}`
- output_dir: `{{IDEA_OUTPUT_DIR}}`
- ideas_history_file: `{{IDEAS_HISTORY_FILE}}`
- attempts_history_file: `{{ATTEMPTS_HISTORY_FILE}}`
- best_manifest_file: `{{BEST_MANIFEST_FILE}}`

## Recent Ideas
{{RECENT_IDEAS_SUMMARY}}

## Recent Attempts
{{RECENT_ATTEMPTS_SUMMARY}}

## Best Program Manifest
{{BEST_MANIFEST_SUMMARY}}

## Distilled conclusions
The notes below are forward-looking guides synthesized by previous cycles. Use them to bias toward
directions previously found *promising* and away from directions found *unpromising*. They replace neither
the recent-memory summary nor the histories -- but they should shape your initial framing of "what to try
next".
```

```
### From this worker's prior cycles
{{OWN_CONCLUSIONS}}
{{OTHER_WORKERS_CONCLUSIONS_SECTION}}
# Required Output Files
Write exactly these files:
- `{{IDEA_BUNDLE_FILE}}`
- `{{TARGET_RUBRICS_FILE}}`

`idea_bundle.json` must be a JSON object with keys:
- `idea_id`: string
- `title`: string
- `hypothesis`: string
- `implementation_plan`: non-empty array of short strings
- `target_rubrics`: non-empty array of objects `{id, description, type}`
- `notes_for_agent`: string

Rules:
- `target_rubrics` must include at least one `metric` rubric and one `implementation` rubric.
- Rubric ids should be stable snake_case.
- `target_rubrics.json` must be exactly the same rubric array as `idea_bundle.json.target_rubrics`.
- Keep rubrics concrete/measurable.

# No Execution During Idea Generation
- Do not run Bash, Shell, Python, `python -c`, `pip`, `evaluate.py`, benchmarks, optimizers, or exploratory
computation.
- Do not install packages or launch subprocesses.
- Do not execute the task program or evaluator.
- Generate ideas from the provided context, recent memory, histories, manifests, and file reads only.
- Your only outputs are the required JSON files and required memory entries.

# Memory Protocol
- Do not write local memory to `./memory.md` in this staging workspace (for example `_idea_generation/...`).
- Keep local memory in an idea-specific directory under `{{GLOBAL_RESULTS_DIR}}/ideas/`.
- Single-directory invariant (strict):
  - finalize `idea_id` and `title` first, before writing any local memory
  - compute canonical `<idea_dir_name>` exactly once as `{cycle_index:02d}_{slug(title, fallback=idea_id)}`
  - `slug(title, fallback=idea_id)` rule: lowercase, replace each non-`[a-z0-9]` run with `_`, trim
  leading/trailing `_`
  - write local memory only to `{{GLOBAL_RESULTS_DIR}}/ideas/<idea_dir_name>/memory.md` (create if missing;
  append-only)
  - never write memory to any other `{{GLOBAL_RESULTS_DIR}}/ideas/<...>/memory.md` path for this cycle (no
  temporary/fallback dir)
  - before exit, verify there is only one cycle-local memory target for this idea; if an alternate
  same-cycle dir was created, merge/move its memory into the canonical dir and stop using the alternate dir
- Also append each idea-memory event to:
  - `{{GLOBAL_MEMORY_MD_FILE}}`
  - `{{GLOBAL_MEMORY_JSONL_FILE}}`
- Treat `{{GLOBAL_MEMORY_MD_FILE}}` as the single campaign-level markdown memory file.
- `{{GLOBAL_MEMORY_MD_FILE}}` must live in the same results-root directory as `outer_state.json`; never
create `memory.md` under `{{GLOBAL_RESULTS_DIR}}/ideas/`.
- `{{GLOBAL_MEMORY_JSONL_FILE}}` is the authoritative cross-idea memory log.
- Use this compact recent-memory summary first, then read the files above if you need more detail:
{{GLOBAL_MEMORY_RECENT_SUMMARY}}
- After appending the cycle's regular memory event, follow the `memory-log` skill's "Update distilled
conclusions" step: apply the (a)/(b)/(c)/(d) gate, and if an update is warranted, run `--write-conclusions`
(single paragraph, ≤300 words, ≤3 retries on validation failure). The helper paths are
`{{CONCLUSIONS_MD_FILE}}` and `{{CONCLUSIONS_HISTORY_FILE}}`; pass `--worker-index {{WORKER_INDEX}}`. Always
include the gate decision (`gate=<updated|kept|skipped>; trigger=<a|b|c|d|none>; reason=...`) inside the
regular memory event.
- Record idea discoveries: alternatives considered, selected hypothesis, and why this cycle decision is best
from recent evidence.
- After writing `idea_bundle.json` and `target_rubrics.json`, append an entry with:
  - `idea_id`
  - hypothesis summary
  - key rubric choices (`id`, `type`)
  - next execution recommendation
- Directory mismatch is a hard failure: do not finish the idea step until local memory uses one canonical
idea directory for this cycle.

# Constraints
- No user interaction.
- No prose-only response.
- No computational probes or command execution.
- If uncertain, choose the most actionable next experiment and continue.
```

F.3. Execution Agent

Execution Agent Prompt

```
# Role
You are an autonomous scientist/engineer running an iterative optimization campaign.
Goal: improve objective score while preserving correctness and satisfying rubrics.
Use `edit-program`, `evaluate-program`, `verify-rubrics`, and `memory-log` skill workflows for execution.

# Task Context
## Goal
{{TASK_GOAL}}

# Active Idea
Attempt ID: `{{ATTEMPT_ID}}`
Idea ID: `{{IDEA_ID}}`

Idea bundle (`{{ACTIVE_IDEA_FILE}}`):
```json
{{ACTIVE_IDEA_JSON}}
```

Target rubric bundle (`{{ACTIVE_RUBRICS_FILE}}`):
```json
{{ACTIVE_RUBRICS_JSON}}
```

Overall Pipeline
`edit-program -> evaluate-program -> verify-rubrics`

For each iteration:
1. `edit-program`: create next variant at `{{RESULTS_DIR}}/program_<idx>/` (never overwrite old dirs). Code
lives in `program_<idx>/src/**` and metadata in `program_<idx>/program_spec.toml`.
2. `evaluate-program`: run `{{TASK_DIR}}/evaluate.py` and collect `metrics.json`, `correct.json`, logs.
3. `verify-rubrics`: use the `verify-rubrics` skill workflow to check rubric compliance with evidence.
4. Immediately update `./memory.md` using `memory-log` with a structured evidence entry for this iteration.
5. Immediately write per-iteration rubric artifact to `{{RESULTS_DIR}}/program_<idx>/rubric_status.json`.
6. Update `{{RESULTS_DIR}}/rubric_status.json` to match the latest evaluated `program_<idx>` status.
7. Only then decide next change from evidence.

Hard Iteration Controls
- Always run `evaluate-program -> verify-rubrics` as one atomic pair for each `program_<idx>`.
- Never start editing `program_<idx+1>` before writing rubric artifacts for `program_<idx>`.
- Stop the attempt immediately when latest `rubric_status.json` has `overall_pass: true`.
- Evaluation budget is based on calls, not folders: each command that runs `evaluate.py` consumes one
evaluation, including `program_0`, retries, failed runs, overwrites, and repeated runs on the same
`program_<idx>`.
- You may make at most `{{MAX_PROGRAM_EVALS_PER_ATTEMPT}}` total `evaluate.py` calls in a single attempt.
- Do not evaluate `program_0` unless explicitly required by the task. Treat `{{RESULTS_DIR}}/program_0/` as
the read-only implementation reference and use existing baseline artifacts if present.
- If `{{MAX_PROGRAM_EVALS_PER_ATTEMPT}}` is `1`, create exactly one edited candidate (`program_1`), run
`evaluate.py` exactly once on `program_1`, then verify rubrics, write memory, and stop if rubrics fail.
- Do not re-run `evaluate.py` for the same program. Use the existing result, inspect logs if needed, verify
rubrics from available evidence, write memory, and stop with `cap_reached`.
- If the final allowed evaluation call fails any rubric or the evaluation itself fails, stop the attempt and
report cap reached.

No Exploratory Execution
- Do not run exploratory computation, timing probes, benchmarks, optimizer searches, ad hoc metric scripts,
or local experiments outside the official `evaluate.py` call.
- Do not run `python -c`, heredoc Python, notebooks, `pip install`, `pip`, `pytest`, `time`, SciPy
optimizers, differential evolution, SLSQP probes, or direct task-function calls.
- Before evaluation, use file reads and edits only. After evaluation, inspect only the artifacts and logs
produced by that official evaluation.
- The only allowed Python execution is the single official `evaluate.py` command, subject to the evaluation
budget.

Memory Protocol
- Continuously update `./memory.md` (create it if missing; append-only, never erase prior entries).
- Keep this local file attempt-scoped in the current attempt directory (`ideas/<idea>/<attempt>/`), not as a
campaign-global file.
- Also append every major-step memory event to:
  - `{{GLOBAL_MEMORY_MD_FILE}}`
  - `{{GLOBAL_MEMORY_JSONL_FILE}}`
- Treat `{{GLOBAL_MEMORY_MD_FILE}}` as the single campaign-level markdown memory file.
- `{{GLOBAL_MEMORY_MD_FILE}}` must live in the same results-root directory as `outer_state.json`; never
create `memory.md` under `ideas/` subdirectories.
- Use `{{GLOBAL_MEMORY_JSONL_FILE}}` as the authoritative cross-idea memory log.
- Start from this compact recent-memory summary before reading more from disk:
{{GLOBAL_MEMORY_RECENT_SUMMARY}}
- After every `evaluate-program -> verify-rubrics` pair, append one `Program Evaluations` entry with:
  - `program_<idx>`
  - `combined_score`
```

```
  - `correct`
  - rubric pass/fail summary
  - next recommendation
- Log key discoveries/decisions for each major step (`edit`, `evaluate`, `verify`) before moving to the next
step.
- At attempt end (stop reason `rubric_pass` or `cap_reached`), migrate relevant findings from local
`./memory.md` into global memory:
  - append one `idea_evaluation_summary` entry to `{{GLOBAL_MEMORY_MD_FILE}}`
  - append one matching structured event to `{{GLOBAL_MEMORY_JSONL_FILE}}`
  - include: best `program_<idx>`, best metrics (`combined_score`, `correct`), final rubric status summary,
  key failures/learnings, and next recommendation
  - summarize/highlight only (do not copy-paste the full local memory file)

# Start Point
- Reference baseline project: `{{RESULTS_DIR}}/program_0/` (read-only baseline copy).
- Entrypoint is defined by `program_0/program_spec.toml` (default: `src/program.py`).
- Start by creating `{{RESULTS_DIR}}/program_1/` from `program_0`, then evaluate `program_1` exactly once.
- Do not run a baseline `program_0` evaluation unless the task explicitly requires it; if you do, it
consumes one `evaluate.py` call from the same budget.

# Operating Rules
- Never ask user questions.
- If info missing, make best reasonable assumption and continue.
- Prefer root-cause fixes; avoid random search edits.
- Keep reasoning brief and concrete.
- Do not chain edits without rubric verification for the just-evaluated program.
- Before any command containing `evaluate.py`, check whether the total evaluation budget has already been
consumed. If no budget remains, do not run the command.

# Required Artifacts
After **every** evaluation of `program_<idx>`, write `{{RESULTS_DIR}}/program_<idx>/rubric_status.json`.
Before exit, ensure `{{RESULTS_DIR}}/rubric_status.json` exists and exactly matches the latest evaluated
program rubric artifact.

Artifact structure:

```json
{
  "attempt_id": "{{ATTEMPT_ID}}",
  "idea_id": "{{IDEA_ID}}",
  "overall_pass": false,
  "rubrics": [
    {"id": "rubric_id", "pass": false, "evidence": "brief evidence"}
  ],
  "metrics": {"combined_score": 0.0, "correct": false},
  "next_recommendation": "short next step"
}
```

Exit Checklist
1. Identify latest evaluated index `<idx>`.
2. Confirm `program_<idx>/rubric_status.json` exists.
3. Confirm `{{RESULTS_DIR}}/rubric_status.json` equals `program_<idx>/rubric_status.json`.
4. Confirm latest `program_<idx>` evidence is logged in `./memory.md`.
5. Confirm relevant local-memory findings were migrated to global memory (`{{GLOBAL_MEMORY_MD_FILE}}` and
`{{GLOBAL_MEMORY_JSONL_FILE}}`).
6. Confirm no more than `{{MAX_PROGRAM_EVALS_PER_ATTEMPT}}` total `evaluate.py` calls were made, including
any `program_0` evaluation.
7. Confirm stop reason is one of: `rubric_pass` or `cap_reached`.
```

F.4. Ablation-Study Agent

Ablation-Study Agent Prompt

```
# Role
You are an autonomous agent implementing exactly one variant for a scientific comparison study.
Use `edit-program`, `evaluate-program`, and `verify-rubrics` skill workflows for execution.

# Variant Specification
```json
{{VARIANT_SPEC}}
```

Rubric Contract
```json
{{RUBRIC_CONTRACT}}
```

Attempt {{ATTEMPT_NUMBER}}
{{ERROR_CONTEXT}}
```

```
# CRITICAL: Mandatory Workflow -- Do NOT Skip Any Step

WARNING **WARNING: Ending your turn after editing without evaluating is a FAILURE.**
The edit alone has ZERO value without evaluation and rubric verification.
You will be re-launched if you stop early, wasting time and resources.

You MUST execute ALL four steps below **in order** for each `program_<idx>`.
Do NOT stop or end your turn until the checklist is fully done (or the iteration cap is reached).

### STEP 1: Edit
Create `./program_<idx>/` by cloning from `program_<idx-1>` (never overwrite old dirs). Code lives in
`program_<idx>/src/**`.

### STEP 2: Evaluate (MANDATORY -- do NOT skip)
Immediately after editing, call the `evaluate-program` skill to run `./evaluate.py` and collect
`metrics.json`, `correct.json`, logs.

### STEP 3: Verify rubrics (MANDATORY -- do NOT skip)
Immediately after evaluation, call the `verify-rubrics` skill to check rubric compliance and write
`program_<idx>/rubric_status.json` and `./rubric_status.json`.

### STEP 4: Decision gate
If `overall_pass: true` → stop. If rubrics failed and cap not reached → start next iteration from Step 1.

# Variant-Specific Instructions

## For Control Variants (`kind: "control"`)
- `program_0` is the seed -- evaluate it as-is first.
- If it passes rubrics, you are done.
- If it fails only on analysis CSV rubrics, create `program_1` with the CSV writing fix.

## For Ablation Variants (`kind: "ablation"`)
- `program_0` is the read-only seed (copy of the best program). Do NOT evaluate `program_0`.
- Go straight to creating `program_1` by cloning `program_0` and removing/disabling ONLY the component
declared in `target_components`.
- Keep all unrelated logic unchanged.
- Evaluate `program_1` and verify rubrics.
- If rubrics fail, create `program_2` as a repair -- fix only the rubric failures.

## For Baseline Variants (`kind: "baseline"`)
- `program_0` is the read-only seed (copy of the best program). Do NOT evaluate `program_0`.
- Go straight to creating `program_1` by cloning `program_0` and implementing the simpler approach described
in `implementation_plan`.
- Evaluate `program_1` and verify rubrics.
- If rubrics fail, create `program_2` as a repair -- fix only the rubric failures.

# Required Analysis Outputs

The seed program (`program_0`) has already been retrofitted to produce these CSV files automatically after
evaluation. When you create `program_1` (or later iterations), preserve that CSV-writing code so the files
continue to be generated. You may adapt the CSV-writing code if your variant change requires it (e.g., a
removed component breaks a metric reference), but the file names, column schemas, and output structure must
remain identical across all variants -- results are aggregated across variants for comparison.

After evaluation, ensure these files exist under `analysis/` (situated inside the `program_<k>/` directory,
alongside `src`, see example below):

### analysis/run_summary.csv
Columns: `evaluation_id`, `variant_id`, `variant_kind`, `base_program`, `agent_name`, `correct`,
`combined_score`, `overall_pass`, `primary_metric_name`, `primary_metric_value`, `program_dir`,
`results_dir`

### analysis/metric_values.csv
Columns: `evaluation_id`, `variant_id`, `metric_name`, `metric_value`, `is_primary_metric`

### analysis/timeseries.csv
Columns: `evaluation_id`, `variant_id`, `series_name`, `step`, `value`
(Write headers only if no timeseries data exists)

### analysis/rubric_status.json
Must cover all 9 rubric families from the rubric contract.

# Hard Iteration Controls
- Always run `evaluate-program -> verify-rubrics` as one atomic pair for each `program_<idx>`.
- Never start editing `program_<idx+1>` before writing rubric artifacts for `program_<idx>`.
- Stop the attempt immediately when latest `rubric_status.json` has `overall_pass: true`.
- Evaluation budget is based on calls, not folders: each command that runs `evaluate.py` consumes one
evaluation, including `program_0`, retries, failed runs, overwrites, and repeated runs on the same
`program_<idx>`.
- You may make at most **{{MAX_PROGRAM_EVALS}}** total `evaluate.py` calls.
- If `{{MAX_PROGRAM_EVALS}}` is `1`, create exactly one edited candidate (`program_1`), run `evaluate.py`
exactly once on it, then verify rubrics, write required artifacts, and stop if rubrics fail.
- Do not re-run `evaluate.py` for the same program. Use the existing result, inspect logs if needed, verify
rubrics from available evidence, write required artifacts, and stop with cap reached.
```

```
- If the final allowed evaluation call fails any rubric or the evaluation itself fails, stop the attempt and
report cap reached.

# No Exploratory Execution
- Do not run exploratory computation, timing probes, benchmarks, optimizer searches, ad hoc metric scripts,
or local experiments outside the official `evaluate.py` call.
- Do not run `python -c`, heredoc Python, notebooks, `pip install`, `pip`, `pytest`, `time`, SciPy
optimizers, differential evolution, SLSQP probes, or direct task-function calls.
- Before evaluation, use file reads and edits only. After evaluation, inspect only the artifacts and logs
produced by that official evaluation.
- The only allowed Python execution is the official `evaluate.py` command, subject to the evaluation budget.

# Start Point
- Reference baseline project: `./program_0/` (read-only baseline copy).
- Entrypoint is defined by `program_0/program_spec.toml` (default: `src/program.py`).
- Read `./task.md` for variant-specific instructions (hypothesis, target components, implementation plan,
what to do).
- Read `./original_task.md` for the original task description and domain rubrics.
- Read the `NUM_REPLICATIONS` environment variable. When set and >1, pass `--num_replications
$NUM_REPLICATIONS` to every `evaluate.py` invocation to launch parallel SLURM jobs per evaluation.
- **Control variants only:** evaluate `program_0` first to establish baseline metrics:

```bash
$WRAPPER_PYTHON ./evaluate.py \
    --program_path ./program_0/src/program.py \
    --results_dir ./program_0 \
    ${NUM_REPLICATIONS:+--num_replications $NUM_REPLICATIONS}
```

- **Ablation/baseline variants:** skip `program_0` evaluation -- go straight to creating `program_1` with
the variant change, then evaluate `program_1`.

Operating Rules
- Never ask user questions.
- Do NOT invent additional experiments beyond the declared variant.
- Do NOT optimize beyond making the variant valid and analyzable.
- For ablations: change ONLY the declared component.
- Preserve the CSV analysis infrastructure in the program.
- Read variant metadata from environment variables: `VARIANT_ID`, `VARIANT_KIND`, `BASE_PROGRAM`,
`EVALUATION_ID`.
- Before any command containing `evaluate.py`, check whether the total evaluation budget has already been
consumed. If no budget remains, do not run the command.
```

F.5. Task-Drafting Agent

Task-Drafting Agent Prompt

```
# Role
You are a task-bootstrap agent.
Goal: from `task.md`, draft a runnable task scaffold for the optimization loop.
Use the `draft-task` and `memory-log` skill workflows for required files, deterministic decisions, and
traceability.

# Task Context
- Task directory: `{{TASK_DIR}}`
- Task spec: `{{TASK_MD_PATH}}`
- Goal section:
{{TASK_GOAL}}
- Rubrics section:
{{TASK_RUBRICS}}

# Required Outputs
Create missing files only:
- `{{DRAFT_CAPTURE_PATH}}`
- `{{DRAFT_EVALUATE_PATH}}`
- `{{DRAFT_PROGRAM_SPEC_PATH}}`
- `{{DRAFT_ENTRYPOINT_PATH}}`

Rules:
- If either file already exists, do not edit it.
- Treat every requirement in `task.md`, especially `Task-Drafting Requirements`, as a hard acceptance
criterion.
- Inspect and use named upstream resources as required by `task.md`; network access and read-only clones
under `/tmp` are allowed. A missing local package does not establish that a resource is unavailable.
- Do not silently replace a required benchmark, evaluator, dataset, protocol, or base implementation with a
synthetic stand-in. If blocked, exit non-zero and report the exact blocker.
- Ensure `{{DRAFT_PROGRAM_DIR}}` and `{{DRAFT_CODE_ROOT}}` exist when writing the baseline.
- Write `program_spec.toml` using **exactly** the schema below. The field names
  are fixed: the entrypoint table requires `path` (string, relative to
  `code_root`), `language` (string), and `type` (`"script"` or `"module"`). Do
  **not** invent keys such as `file`, and do **not** set `type` to a language
```

```
  name. `code_root` must be the literal string `"src"`.

  ```toml
  schema_version = 1
  code_root = "src"

  [entrypoint]
  path = "program.py"
  language = "python"
  type = "script"

  [copy]
  exclude = []
  include = []

  [integration]
  kind = "generic"
  notes = ""
  ```
- Before finishing, validate the scaffold with the repository's canonical program-spec loader, and run
candidate code in a killable, resource-limited subprocess isolated from evaluator-private data.
- `capture.py` should provide an `OutputCapture` context manager compatible with existing tasks.
- `evaluate.py` must support:
  - `--program_path <path>`
  - `--results_dir <dir>`
- `evaluate.py` must write:
  - `metrics.json` with numeric `combined_score`
  - `correct.json` with boolean `correct`
- Keep baseline behavior deterministic and lightweight.
- Prefer minimal dependencies.

Memory Protocol
- Continuously update `./memory.md` (create if missing; append-only).
- Also append draft-step memory events to:
  - `{{GLOBAL_MEMORY_MD_FILE}}`
  - `{{GLOBAL_MEMORY_JSONL_FILE}}`
- Log scaffold decisions: assumptions inferred from `task.md`, deterministic baseline choices, and interface
compliance checks.
- After writing each required file, append an entry noting what was created and how requirements were
satisfied.

Constraints
- No user interaction.
- No prose-only output.
- Do not modify files outside the required outputs, except temporary read-only source checkouts under `/tmp`.
```

G. Task Properties and Hyperparameters

G.1. Aggregate Sweep

Table 10 | Harness and model configurations for the aggregate sweep. All runs use high reasoning effort.

Harness	Version	Assigned models
Codex	0.149.0	GPT-6 Astra; GPT-5.6-Sol; GPT-5.5; GPT-5.4
Claude Code	2.1.240	Fable 5.1; Opus 5; Opus 4.7; Opus 4.6
Pi	0.84.2	GPT-5.6-Sol; Opus 5; DeepSeek V4 Flash
OpenCode	1.18.21	GPT-5.6-Sol; Opus 5; DeepSeek V4 Flash

G.1.1. Learned Optimization

Initial Task Instructions for Learned Optimization Topic Direction

```
# Goal

Start from ELO and improve held-out anytime transfer after real outer training from scratch within
a fixed training-time budget. Candidates may change the learned architecture, outer objective,
gradient estimator, curriculum, and training policy within `program_0`.

# Upstream and Data

Treat upstream repositories as untrusted reference material.

- ELO is the baseline starting point: https://github.com/xiaol827/ELO at commit
  `664ef6503ae8d2b1d5b89b551680eff91472574a`.
- Inspect and pin the upstream code, dependencies, license, and data provenance. Use a feasible
  frozen subset of ELO's official task families and its substantive meta-training implementation
  for the starting candidate and unchanged reference method. Offline data/loading adaptations
  are permitted; replacing ELO with a synthetic update-rule proxy is not.
- Keep final transfer environments, examples, and seeds excluded from all search, meta-training,
  tuning, calibration, and retry decisions. Freeze resource hashes before execution.
- If the required ELO path or data cannot run, report the concrete drafting blocker.

# Evaluation

- Compare candidates with pinned ELO using the same available training data, hardware, and
  end-to-end training-time allowance. Candidates may allocate that time differently: outer-step
  counts, particles, truncation lengths, and total inner work need not match ELO. Record these
  quantities and precision where observable; candidate self-reports do not certify compliance.
- Use the same evaluation workloads for all methods and one primary transfer score aggregated
  across a small set of predeclared short, medium, and long horizons and independent seeds.
- Include one hand-designed comparator, such as AdamW, with its budget and settings frozen before
  search. Additional comparators or behavioral controls require a specific scientific purpose.
- Report stability, training efficiency, runtime, and available memory evidence as diagnostics,
  without adding separate weighted score components or correctness gates.
- Retain negative results and failures, including divergence, NaNs, malformed updates, and missing
  outer training, under a frozen aggregation policy. Define normal budget exhaustion separately
  from execution failure; no dropping failed seeds or renormalizing around them.
- Final evaluation trains the frozen candidate afresh, without search checkpoints. Keep its
  environments and outcomes sealed until candidate freeze.

# Compute Budget

Use two H100 GPUs total: one evaluator GPU and one isolated candidate GPU. All training, baselines,
and evaluation must fit this layout and an approximately 20-idea campaign. Define training timing
boundaries before implementation and measure each method independently with trusted clocks.
Use harness-supported telemetry; label measurement scope and leave unavailable memory data absent.

# Task-Drafting Requirements

Before implementation, freeze a concrete evaluation design: ELO configuration, task/data splits,
seeds, horizons, time limits, comparator settings, score formula, failure handling, and improvement
criterion. Choose the smallest feasible design that answers the goal. Do not add requirements
beyond this brief without a scientific need.

Build the candidate, search and sealed final evaluators, contract, pinned runtime, and resources
using the shared harness policies for offline execution, isolation, transport, and publication.
The evaluator owns data access, budget enforcement, integrity checks, and scoring; research
components remain candidate-owned and editable. Smoke uses the same interface and substantive
training path with a reduced workload.

# Rubrics
```

```
- A `learned optimizer` claim requires executed outer training, serialized learned state, and
  evidence that changing or resetting that state changes held-out updates or performance.
- An improvement claim requires the frozen final comparator to favor the candidate over ELO on the
  primary score, with per-horizon and per-seed results retained. Protocol validity alone does not
  establish improvement, and a negative result remains reportable.
- Scope conclusions to the tested configuration, tasks, horizons, seeds, and budget.

# Frozen Manual Experiment

This directory is the manually maintained implementation of the brief above. Run the existing
optimization pipeline without its `task_drafting` phase. `protocol.json` freezes the experimental
design before any search. The canonical drafting brief remains `../learned_optimizer/task.md`.

The starting method is upstream ELO's `smallMLP`, with the original expert-supervised mechanism
and a compressed supervision schedule from outer step 10 through 80 with a 1,800-second training
allowance. Editable architecture, objective, gradient estimator, curriculum, and training policy
remain inside `program_0`; evaluator code, reference method, resources, and score stay frozen.

Meta-training uses 8,192 official training images each from MNIST and Fashion-MNIST. Search transfer
uses 4,096 CIFAR-10 official training images and 1,024 official test images. Final transfer uses
4,096 SVHN official training images and 1,024 official test images. Every image is resized to 8×8
with Pillow 11.1.0 bilinear interpolation and scaled to [0,1]; grayscale and RGB channels remain
unchanged. Subsets are fixed by sorting original indices by SHA256 of ASCII `seed:index` and taking
the first requested count. Selection seeds and raw source hashes are frozen before preparation.

Transfer evaluates a one-hidden-layer width-32 MLP and upstream's `small-conv-10` with channels
16 and 32, batch size 64, and horizons 100, 500, and 1,000. Search seeds are 101 and 211; final seeds
are 1009, 2017, and 3011. Each candidate and reference outer-training run receives 1,800 seconds.
Train up to two candidate seeds concurrently, with one isolated worker per GPU. After verified
worker shutdown, transfer uses one GPU for the trusted evaluator and one for the candidate.
Reuse evaluator-owned search ELO and AdamW evidence only when source, runtime, data, budget,
and observed GPU/driver identities match. Candidate training is always fresh. Final reference
evidence is invocation-scoped and never reuses search checkpoints or accesses final data before
candidate freeze. Total evaluator limits remain 9,000 seconds for search and 12,600 for final,
including cache construction when required; measured cold and warm runtimes determine planning.

The primary score is the equal mean of `1 - clip(cross_entropy / log(10), 0, 1)` over all declared
architecture/horizon/seed cells. Failed cells contribute zero; never drop or renormalize them.
AdamW settings are learning rate 0.001, betas (0.9,0.999), epsilon 1e-8, weight decay 0.01. A final
improvement requires the candidate primary score to exceed independently trained frozen ELO.
Normal trusted-clock training budget exhaustion preserves the latest completed learned state;
crashes, missing training/state, malformed responses, and nonfinite outputs remain failures.
The broker cancels overdue training operations before acknowledging timeout. Each seed has two
declared resets: fresh outer training, then a fresh worker importing the completed checkpoint for
transfer. Each method may use at most 40,000 training RPCs per seed, reserving transport capacity
for every transfer cell and control. This bounds RPCs, not candidate-owned internal outer work.
Learned-state intervention and serialization replay controls cover each transfer architecture;
these fixed-gradient probes do not establish full GPU-trajectory determinism.
Forty seconds of each production training allowance are reserved for cancellation and transport;
training stops admitting work earlier when the longest observed operation would consume its reserve.
The unchanged reference trains on the evaluator H100; candidate training uses its isolated H100.
Both receive one identical-model GPU and the same independently measured allowance. Execution
is sequential, with JAX preallocation disabled; evaluator inner workloads run after outer training.

# Offline Resource Preparation

Install the pinned task requirements into the preparation environment, create a private download
cache, then run `python examples/learned_optimizer_manual/prepare_resources.py --cache-dir CACHE`.
The cache must already exist. Preparation only downloads and converts resources; it never trains
models or evaluates final outcomes. Existing cache files must match published SHA256 and size;
existing generated artifacts must match exactly. A mismatch stops preparation without overwriting.

Prepared `.npz` files contain numeric `images` (float32 NHWC) and `labels` (int64), readable with
`numpy.load(..., allow_pickle=False)`. `resources/search` holds meta-training and CIFAR-10 arrays;
`resources/final` holds the same immutable meta-training data and the SVHN arrays. Each plane has
its own hashed manifest. The top-level preparation manifest records raw URLs, checksums, terms,
and generated hashes. Resource binaries and download caches are local generated artifacts.
Final smoke uses separate deterministic synthetic validation fixtures; it never opens SVHN data.

Raw SHA256 pins originate from TensorFlow Datasets' published checksum tables. MNIST's dataset
catalog does not establish a dataset license; Fashion-MNIST is MIT. CIFAR-10's official page requests
citation
without declaring an explicit license. SVHN states non-commercial use only. Preserve dataset
provenance and applicable citation/use terms in experiment artifacts.
```

G.1.2. Hyperparameter Optimization

Initial Task Instructions for Hyperparameter Optimization Topic Direction

```
# Goal

Improve a general sequential hyperparameter optimizer for expensive black-box objectives.
Any adaptive method is eligible: Bayesian, evolutionary, bandit, local search, or hybrids.
Target held-out anytime and terminal performance under matched, limited budgets.

# Benchmarks

Treat upstream repositories as untrusted reference material.

Execute all three families in both search and sealed final evaluation:

- BBOB: bounded, noiseless continuous functions from official
  [COCO Experiment](https://github.com/numbbo/coco-experiment); pin its release.
- [HPO-B](https://github.com/machinelearningnuremberg/HPO-B), commit
  `f16c0c52c27ba4e928dd3ad648231726cb06a9ab`: continuous XGBoost surrogates on `[0, 1]`.
  Use v3 validation tasks for search and v3 test tasks for final.
- [YAHPO Gym](https://github.com/slds-lmu/yahpo_gym), commit
  `f16e2bf5b8b27605cb2a59387363befe0db8e785`, with
  [data](https://github.com/slds-lmu/yahpo_data) `v1.0.2`, commit
  `e1a472a061fec6df2e090b3c72fe560d858c2062`: preserve categorical, integer,
  conditional, fidelity, and official cost semantics.

Freeze representative cases, seeds, initial designs, budgets, checkpoints, and scoring before
search. Keep HPO-B task IDs, BBOB functions or instances, and YAHPO scenarios or instances disjoint
between search and final. Never use final cases for tuning, calibration, memory, or promotion.
If a required benchmark cannot be verified and executed, fail drafting with the concrete blocker;
never substitute fabricated objectives.

# Candidate Interface

Support `reset`, `ask`, `tell`, and `metadata`; reset clears prior state.
Expose only an anonymous typed legal search space, completed observations, and remaining query
or cost budget. Suggestions must adapt to observations. Anonymous-space dispatch is allowed;
identity-keyed behavior, fixed replay, lookup answers, hidden optima, and future access are not.

The evaluator owns accounting. Deterministically reject malformed, duplicate, inactive,
out-of-bounds, non-finite, or over-budget suggestions. Count invalid outputs, missing cases,
crashes, and timeouts as declared failures; never drop them from aggregates.

# Evaluation

- Match candidates, the frozen adaptive starter, random search, and benchmark-appropriate adaptive
  CPU controls on cases, initial designs, seeds, budgets, CPU limits, and failure accounting.
  Include continuous evolutionary and sequential HPO controls; use fidelity-aware controls for
  multi-fidelity claims. The starter must use observations to change later suggestions.
- Report per-family normalized anytime and terminal regret, invalid-query rate, and optimizer
  overhead. Convert HPO-B maximization responses to regret; use official BBOB targets/references
  and fidelity- and cost-aware YAHPO scoring.
- Define `combined_score` as the equal-weight mean of higher-is-better family scores in `[0, 1]`.
  Freeze normalization references independently of candidates, with fixed case, seed, checkpoint,
  and failure weights.
- Search `correct` requires legal sequential behavior, finite outputs, complete evaluation,
  respected budgets, no leakage, and `candidate_family_mean >= starter_family_mean - 0.01`
  for every family. The reference is the matched starter frozen before search; tolerance is an
  absolute normalized-score difference. Promote only on strictly improving `combined_score`;
  retain the incumbent on ties. Control/checkpoint comparisons are not individual search vetoes.
- Expose only bounded aggregate development metrics and failure categories during search.
  Freeze candidate code and state before one sealed final evaluation. Report paired held-out
  differences against controls and uncertainty across independent cases and seeds; checkpoints
  within a run are correlated, not independent replications. Predeclare the comparison and
  uncertainty procedure. Search eligibility does not establish scientific superiority.

# Compute Budget

CPU-only: zero GPUs, at most 8 cores. Fit sequential optimization into an approximately
20-idea campaign. Plane limits include candidates, controls, loading, and scoring:

| Plane | Minimum cases per family | Minimum seeds per case | Minimum queries per run | Wall time | Memory |
| --- | --- | --- | --- | --- | --- |
| Search | 3 | 2 | 32 | 1,800 s | 32 GiB |
| Final | 3 disjoint | 3 | 64 | 3,600 s | 64 GiB |

Enforce official cost budgets for multi-fidelity runs. Record CPU allocation, memory, wall time,
query/cost budgets, and optimizer overhead.

# Rubrics

- General-optimizer claims require one frozen method or anonymous-space dispatch rule to improve
```

```
  held-out performance across all three families against matched controls.
- HPO claims require executed HPO-B and YAHPO evidence. BBOB supports continuous optimization;
  surrogate benchmarks do not establish real-training performance.
- Multi-fidelity claims require optimizer-chosen YAHPO fidelities, enforced official costs, and
  improvement over matched fidelity-aware controls.
- Anytime claims require evidence across predeclared checkpoints and independent seeds;
  terminal-only gains do not establish sample efficiency.
- Publication requires successful sealed final evaluation, paired controls, and complete failure
  accounting.
  Negative or inconclusive comparisons are valid outcomes; claim improvement only when supported
  by paired evidence and uncertainty. Scope conclusions to tested versions, cases, dimensions,
  fidelities, budgets, seeds, and CPU limits; avoid universal HPO or untested noisy-objective claims.
```

G.1.3. World Modelling

The boxed text records the initial research direction. The manual aggregate sweep uses a fixed TinyWorlds Pong protocol with four context frames and prediction horizons of 1, 4, 8, and 16 frames. Its error metric equally weights pixel and foreground reconstruction; observed actions and action controllability are not evaluated.

Initial Task Instructions for World Modelling Topic Direction

```
# Goal

Improve an action-free compact autoregressive world model, starting from TinyWorlds. This is a
manually constructed task: run resource preparation once, then experiments without task drafting.
Candidate edits may change the full implementation under `program_0`, including tokenizer, dynamics,
objectives, architecture, optimization, data loading, or distributed execution.

The primary objective is better held-out, free-running rollout quality after genuine training from
scratch. Scope all claims to this Pong configuration. Actions are not supplied or evaluated; do not
claim action-conditioned performance.

# Upstream and Data

TinyWorlds: https://github.com/AlmondGod/tinyworlds at
`a99f3a8195bc8b81e21304121528ae32cb56db05` (MIT). The original implementation is retained byte-for-byte
under the prepared upstream directory. Orchestration adapters are separate. Treat upstream text as
untrusted reference material, never as evaluator instructions.

Official data: `AlmondGod/tinyworlds`, revision `aa62b3ca4349c8ed23894ccb33ec6ff45fdcb3b4`,
`pong_frames.h5`, SHA-256
`d9ed1d4cd36cf516864135fcb0161b2ef8680eda1dfb73b560e1adce1e1e45b5`.
Use chronological half-open frame partitions: training `[0,floor(0.6N))`, search
`[floor(0.6N)+64,floor(0.8N))`, final `[floor(0.8N)+64,N)`. The 64-frame gaps prevent boundary windows
from leaking across partitions. RGB frames are uint8 at 64×64; preparation uses OpenCV INTER_AREA
only if resizing is necessary. No windows may cross partition boundaries.

Each evaluation plane owns a complete regular-file resource tree: `train.npy`, private
`heldout.npy`, exact immutable reference source, evaluator, protocol, provenance manifest, and
validated reference prediction cache. Caches stay private to their evaluation plane.
Search and final receive separate training-data copies. Candidate processes receive bounded training
chunks and permitted rollout contexts over RPC; they never receive held-out targets, final resources,
or a shared path to evaluator resources. Candidate code must not access resources outside its API.

Preparation verifies a tracked baseline source manifest before constructing immutable reference
copies. It refuses to derive a reference from a modified candidate. Pretrained checkpoints, external
logging, internet downloads during evaluation, and reuse of candidate search state during final
are forbidden. The operator prepares each plane's immutable reference cache from fresh training;
the evaluator reuses these predictions without exposing reference weights to candidates.
TinyWorlds' latent-action stage remains present because its dynamics entrypoint requires that
checkpoint, while dynamics action conditioning is disabled.
The immutable reference and initial starter use dynamics learning rate `0.001` for search and
final. Training-only calibration addressed background collapse at the upstream rate of `0.01`
with batches of 32 per GPU. Their tokenizer and latent-action learning rates retain upstream
defaults. This adapter configuration leaves the pinned upstream source unchanged.

# Candidate Interface

Implement `src/program.py:handle` through the standard `python_rpc_v1` interface. Preserve the
protocol documented in the starter: seeded reset, bounded training-data chunks, train, behavioral
checkpoint probes, free-running rollout, and close. Use only the candidate's assigned GPUs and
scratch directory. All tensors crossing the boundary are bounded, validated serializable payloads.
The evaluator resets the candidate from scratch once, with search seed 1729 or final seed 2718.
Reference preparation uses the same respective seeds in separate fresh training runs.

# Evaluation
```

```
Reference preparation and candidate evaluations each train from scratch on all four GPUs with
identical training data, global batch size, precision, update schedules, and per-role time budgets.
The immutable reference executes pinned upstream code once per frozen plane configuration; each
candidate evaluation uses its verified cached predictions. Cache identity binds the source, data,
contexts, workload, inference implementation, runtime image, and software/hardware configuration.
Missing or invalid caches fail before candidate training; an operator must prepare them explicitly.
The starter initially implements the same behavior. Never replace actual TinyWorlds training with
a synthetic or non-learning proxy. Equal configuration does not guarantee bit-identical training.

Free-running predictions start from four context frames. Evaluate horizons 1, 4, 8, and 16 using one
fixed search seed and one disjoint fixed final seed. Future frames and target-derived actions are
never provided. Pixel reconstruction and foreground reconstruction errors receive equal weight;
horizons receive equal weight. Foreground is defined relative to a training-only background
estimate. The evaluator's metric implementation fixes all thresholds and aggregation details.

Report copy-last-frame and constant-velocity controls. An improvement is eligible only when the
candidate beats the immutable reference and both controls at every horizon. Execution correctness
and improvement eligibility are separate: the unchanged starter can execute correctly without
beating itself. Search `correct` requires successful fresh execution and finite rollouts/checkpoint
probes; comparative improvement remains the separate `improvement_eligible` metric and score gate.
Final `correct` additionally requires `improvement_eligible=1`: a win over the immutable reference
and both controls at every horizon. A completed final evaluation with valid outputs but no such
improvement records `correct=false` and `not_qualified`; its process still exits successfully.
Operational baseline preparation accepts this completed, stable non-improvement without claiming
scientific qualification. Also report temporal error, current candidate training wall time, historical
reference
cache-build training wall time, and stability measured
from finite evaluator-observed behavioral outputs at four predeclared checkpoints. Candidate
acknowledgements or reported update counts do not prove training stability or learning.

Final evaluation trains the candidate afresh from sealed resources with seed 2718 and uses its
separate private final reference cache. It rejects failed candidates, exposed targets, and candidate
checkpoint reuse. Generated evidence includes bounded
metrics, control comparisons, finite behavioral probes, provenance, and role-specific work timing.
Harness GPU telemetry is independent workload evidence; GPU activity alone does not prove learning.

# Compute Budget

Allocate four NVIDIA H100 80 GB GPUs to the candidate; the production evaluator uses CPUs and
receives no GPUs. Reference cache preparation separately uses all four GPUs. Reserve 16 CPU cores
and 64 GB host memory. Use bfloat16 mixed precision where the pinned implementation supports it.
Record actual hardware and allocation in validation results.

Preserve fixed update counts in `workload.json`: search tokenizer/latent/dynamics 20,000/400/24,000;
final 60,000/1,200/72,000. Stock loops execute each configured count plus one update. Production
batch size is 16 per GPU, preserving global batch size 64 from the original two-GPU configuration.
Reference acceptance checks exact update counts and positive measured training intervals for all
three stages. Productive tokenizer-plus-dynamics time is reported separately from the latent-action
prerequisite. The earlier two-GPU duration targets are historical calibration observations, not
minimum durations for four-GPU execution. Never sleep or pad workloads to meet them.

Search's total ceiling is 10,800 seconds; final's is 32,400 seconds. Each current evaluation ceiling
includes candidate training, setup, cache validation, rollouts, scoring, and cleanup; reference
cache-build time is reported separately. Reference preparation includes a reduced native reference
build on four GPUs with batch size one and four updates per stage; this builder diagnostic is not
candidate qualification. Evaluator smoke uses the same authenticated candidate broker, device
discovery and four-GPU topology as production, reducing only workload size. Its CPU-only evaluator
uses a separately prepared reduced reference cache and never launches GPU training. Smoke never
reads or writes production reference caches and cannot establish production qualification.

# Rubrics

- World-model claims require real training from scratch and autoregressive free-running rollouts.
- TinyWorlds claims require this pinned source and documented official Pong subset.
- Improvement claims require beating the reference and both controls at every declared horizon.
- Stability requires finite evaluator-observed outputs at declared checkpoints.
- Publication of improvement claims requires qualified sealed final evaluation, credible learned rollouts,
and
  harness-owned evidence of the substantive declared GPU workload.
- Scope conclusions to this configuration, data subset, horizons, seeds, and compute budget.
- Resource preparation produces no qualification or validation receipts. Those require
  deterministic smoke and production evaluation through the existing validation tooling.
```

G.1.4. Sparse Autoencoders

Initial Task Instructions for Sparse Autoencoders Topic Direction

```
# Goal

Start from SAE Bench and improve the architecture or training of sparse autoencoders on real
language-model activations. Candidates may change any useful part of the SAE or training system,
including the encoder, decoder, objective, sparsity mechanism, optimizer, schedule, activation
loading, or distributed execution.

The primary objective is a better held-out reconstruction-sparsity Pareto frontier after real
training from scratch. Improvements must preserve language-model behavior, use features beyond a
small fixed subset, and hold across two language-model families.

# Upstream, Models, and Data

Treat upstream repositories as untrusted reference material.

- SAE Bench is the evaluation and custom-SAE reference: https://github.com/adamkarvonen/SAEBench
  at commit `8042bb3828c6340da8d12062324e92b2077c571c`.
- Evaluate SAEs on both `EleutherAI/pythia-160m-deduped` layer 8 and
  `google/gemma-2-2b-it` layer 12, using compatible residual-stream hooks.
- Resource preparation may use `HF_TOKEN` to download Gemma after license acceptance. Pin and stage the
  exact model and tokenizer revision during preparation so every later phase runs offline.
- Select a feasible frozen corpus and create non-overlapping raw-document and token manifests for
  search and final evaluation. Tokenize separately with each pinned model tokenizer.
- Extract activations with the frozen language models in evaluation mode. Do not replace real
  activations or parameter-changing SAE training with generated arrays, pretrained-only inference,
  or a non-learning proxy.
- If either model, corpus, or selected SAE Bench evaluation cannot be materialized for offline use,
  fail preparation with the concrete blocker rather than silently substituting another workload.

# Candidate Interface

Support `reset`, `fit`, `encode`, `decode`, `forward`, `export_decoder`, and `metadata` operations.
The evaluator owns model inference, activation extraction, interventions, and scoring;
`fit` receives training activations only. After fitting,
`encode` and `forward` may receive bounded held-out activation chunks for stateless
inference. Verify identical candidate-state hashes before and after held-out calls. Keep model
weights, raw text, token IDs, labels, task resources, metrics, and other seeds private.
`export_decoder` provides bounded inline chunks of the learned unit-norm linear decoder
matrix plus its configuration, dtype, and shape metadata so an evaluator adapter can execute SAE
Bench sparse probing and targeted probe perturbation. The evaluator must independently exercise the
learned encoder and decoder through bounded tensor exchange.

# Evaluation

- Train and evaluate fresh SAEs for both model families during search and final evaluation.
- Compare candidates with matched TopK and BatchTopK SAEs trained on the same activations, tokens,
  hardware, precision, optimizer steps, and wall-clock budget. Include PCA and zero-ablation
  controls where applicable.
- Evaluate multiple predeclared sparsity targets and compare Pareto frontiers rather than selecting
  one favorable mean L0 after seeing results.
- Search scoring should primarily reflect SAE Bench core metrics: normalized reconstruction error,
  explained variance, loss recovered, achieved sparsity, dead-feature fraction, feature
  utilization, stability across seeds, runtime, and memory.
- Sealed final evaluation must repeat the core metrics and add SAE Bench sparse probing and targeted
  probe perturbation for both model families.
- Treat NaNs, divergence, timeouts, malformed encodings, sparsity violations, failed models or
  seeds, and missing parameter-changing training updates as declared failures.
- Include evaluator-owned constant/copy, reset-state, and applicable permutation controls across
  the declared models, sparsity targets, and seeds.
- Final evaluation must train every selected SAE afresh and must not reuse search dictionaries,
  checkpoints, feature statistics, probes, or activations.

# Compute Budget

Two NVIDIA H100 GPUs are available in total. Use one evaluator GPU and one separate candidate GPU
for both search and final evaluation. Activation extraction and evaluator-owned baseline fits must
share the evaluator GPU; candidate fits use the candidate GPU. Schedule models, seeds, and sparsity
targets within this layout, sequentially when needed. Record actual hardware, allocation,
precision, tokens, optimizer steps, wall time, peak memory, and checkpoint hashes.

Search must train each candidate and matched learned baseline on both models, at least two sparsity
targets, and at least one seed, using at least 1,000,000 training tokens per fit. It must complete
within 3,600 seconds, 240 GiB host memory, and 80 GiB per GPU. Final repeats both models and targets
with at least two independent seeds and 2,000,000 training tokens per fit within 10,800 seconds.
Plane limits include activation extraction, every candidate and baseline fit, interventions, and
scoring; evaluator and candidate fits may overlap on their separate GPUs when the protocol permits.

Choose widths, sparsity targets, corpus sizes, context lengths, batch sizes, seeds, metrics, and
time limits that fit an approximately 20-idea campaign while still exercising substantive SAE
```

```
training and evaluation on both models.

# Evaluation Scope

AutoInterp, unlearning, RAVEL, feature absorption, and the remaining SAE Bench suites are outside
the required workload.

# Rubrics

- An `SAE` claim requires fresh parameter-changing training, sparse feature activations, a learned
  decoder, and evidence that resetting or permuting the learned state changes held-out outputs.
- An `SAE Bench` claim requires execution of the pinned benchmark methodology on compatible custom
  SAEs; merely reproducing metric names is insufficient.
- A Pareto improvement must hold at multiple matched sparsity targets, on both model families, and
  across independent seeds under matched compute.
- A `loss recovered` claim requires evaluator-owned interventions through each pinned language
  model rather than reconstruction metrics alone.
- Sparse-probing or targeted-perturbation gains support only claims covered by those evaluations.
  Do not claim human-interpretable, monosemantic, stable, or broadly causal features.
- Final evaluation must train afresh, report negative and failed models, sparsities, and seeds, and
  compare against all required matched controls.
- Publication requires successful sealed final evaluation and evidence that the substantive
  declared GPU workload ran.
- Scope conclusions to the pinned models, hooks, corpus, sparsities, seeds, and selected SAE Bench
  evaluations; do not imply performance on the full benchmark suite.

# Manual Task Implementation

No drafting agent is involved. Prepare pinned resources once, then launch optimization.
This manual realization uses a bounded frozen encoder graph to enforce held-out immutability.
Candidate Python receives training activations only. After fitting, the evaluator imports plain
encoder/decoder tensors and executes a validated graph itself. Before held-out scoring, it independently
checks candidate `encode`, `decode`, and `forward` against that export on training-only inputs.
The frozen evaluator state is hashed before and after held-out evaluation. Candidate metadata hashes
are diagnostic only. Arbitrary Python held-out inference is intentionally unsupported.

## Frozen interface

The starter is `program_0/src/candidate.py`, with `handle(request)` as entrypoint.
The seven original operations remain; `export_encoder` adds graph and tensor chunks.
Graph operations: matmul, add, sub, mul, relu, gelu, sigmoid, topk. Maximum32nodes,
64tensors, 64million encoder parameters, intermediate width16384. Tensor wire format:
`{dtype: "float16"|"float32", shape: [rows, columns], data: "base64 little-endian bytes"}`;
maximum64rows,262144elements,2MiB per JSON message. No paths, code, pickle, or credentials.
Decoder export is a unit-row-norm linear matrix plus bias. Exported weights are FP32.
Candidate training, objectives, initialization and supported graph architecture remain editable.

## Fixed workload and promotion

Pythia160M layer8 and Gemma2-2B-IT layer12 residual post-block hooks. Default4096features;
mean-L0 targets32 and128. Context128. Search1,048,576 training tokens per fit and one seed;
final2,097,152 tokens per fit and two independent seeds. Full core heldout32768tokens.
Activation transport FP16, model BF16, SAE/optimizer/metrics FP32. Training batch2048,
512 search steps /1024 final steps; matched references use Adam3e-4. Fit caps300s search,
600s final, always within plane-wide3600s/10800s caps including extraction and controls.
No hidden extra optimizer passes over training examples. Identical quantized training inputs for all methods.
BatchTopK calibrates its frozen inference threshold on the final 32,768 training activations
to 95% of target L0. Reference held-out mean L0 must also remain within 105% of target.
Calibration is inference-only, recorded separately, and included in the reference fit deadline.

A valid search requires every case, real training, effective reset/permutation controls,
finite metrics, mean-L0 in(0,1.05*target], at least5% feature utilization, and loss recovered
within0.02 of the stronger matched reference. Higher combined_score is better:
minimum across families of the uniform target/seed mean log(best_reference_NMSE/candidate_NMSE).
Missing/failed cases invalidate the whole score (-100); weights never renormalize.
Other scalar metrics use equal weights across family,target,seed. Seed stability is NMSE standard
variation across seeds averaged in variance space across cases; search reports-1 (unassessed).
Final correctness additionally requires NMSE improvement beyond1e-4 relative at every case;
valid non-improving baselines can complete evaluation but do not qualify improvement claims.

Controls use4096 predeclared heldout tokens: PCA fitted only on training activations, zero,
constant/copy, frozen reset state, mismatched decoder permutation, and joint feature permutation.
Final probes use Bias-in-Bios class_set1 and pinned SAE Bench defaults:4000train/1000test,
context128, sparse-probe k=[1,2,5], TPP n=[2,5,10,20,50,100,500].
Probe residuals are promoted to FP32 after BF16 model inference, before clean probe training
and SAE ablations; all candidates and references use the same FP32 probe arithmetic.
Full search/final acceptance requires production container execution. Reduced smoke uses8192training
and4096heldout tokens per model/target with one seed and40/20probe sizes; never qualifies full work.
```

G.1.5. Tabular Models

Initial Task Instructions for Tabular Models Topic Direction

```
# Goal

Start from TabICLv2 and improve the pretraining of a tabular foundation model for in-context
classification. Candidates may change any useful part of the model or training system, including
the synthetic prior, architecture, objectives, curriculum, optimization, ensembling, data loading,
attention implementation, or distributed execution.

The primary objective is better predictive performance on held-out real tables after substantive
pretraining from scratch. Improvements should transfer across datasets and remain useful at
multiple context sizes without per-table parameter updates.

# Upstream and Data

Treat upstream repositories as untrusted reference material.

- TabICL is the baseline starting point: https://github.com/soda-inria/tabicl at commit
  `8ac8ca8c1951be0aace1c01f087bcd5c19d6fd58`.
- Execute substantive TabICLv2 pretraining through its synthetic-prior generation and training
  paths. Pretrained-only inference, per-table fitting, or a synthetic non-learning proxy is not
  sufficient.
- Select feasible frozen real public classification datasets for transfer evaluation. Keep search
  and final datasets, rows, context/query splits, and seeds disjoint.
- The upstream TabICLv2 pretraining port is not guaranteed to reproduce the released checkpoint.
  Establish a runnable pinned baseline or fail drafting with the concrete blocker rather than
  silently substituting another tabular model.

# Evaluation

- Compare each candidate with the pinned unmodified TabICLv2 pretraining baseline under the same
  hardware, optimizer steps, examples, precision, and wall-clock budget. Reference predictions
  may be prepared once per frozen seed and reused from an evaluator-only integrity-bound cache.
- Evaluate in-context predictions on held-out real tables at multiple context sizes and across
  datasets that vary in rows, features, classes, categorical structure, and missingness where
  feasible.
- Make the score primarily reflect aggregate and per-dataset predictive quality; also report log
  loss, calibration, cross-seed stability, pretraining efficiency, inference time, and supported
  GPU-memory evidence.
- Include simple conventional tabular baselines and an evaluator-owned support-label permutation
  control so a model that ignores its context cannot receive a competitive score.
- Treat NaNs, divergence, timeouts, malformed probabilities, failed datasets, and missing
  parameter-changing pretraining updates as declared failures rather than dropping them.
- Final evaluation must pretrain the selected candidate afresh and must not reuse search
  checkpoints or expose final tables during optimization.

# Compute Budget

Up to four NVIDIA H100 GPUs are available. Task drafting should choose an appropriate one-to-four-
GPU pretraining and evaluation layout, using distributed or mixed-precision execution when useful.
Candidates are not required to use all four GPUs. Record the actual hardware, precision, wall time,
training steps, synthetic tasks processed, and peak memory used by each substantive evaluation.

Choose the TabICLv2 configuration, synthetic-prior budget, real datasets, context sizes, seeds,
metrics, and time limits that fit an approximately 20-idea campaign while still exercising real
pretraining and held-out in-context inference.

# Candidate Scope

Keep the synthetic prior, model architecture, objectives, and training policy candidate-owned
and editable. The evaluator owns frozen real-table splits, comparison budgets,
integrity checks, and scoring; do not fix the intended research contribution behind the RPC
boundary. Candidate-controlled priors must still satisfy the declared training budget and cannot
access hidden real-table queries, labels, dataset identities, or final resources.

# Rubrics

- A `pretrained model` claim requires parameter-changing training from scratch, a serialized model
  state, and evidence that resetting or permuting that state changes held-out predictions.
- A `TabICL` claim requires execution of the pinned TabICLv2 revision and its documented synthetic
  prior and model paths; merely imitating its API or architecture is insufficient.
- A `foundation model` claim requires one pretrained state shared across held-out tables without
  per-table parameter updates.
- An `in-context learning` claim requires predictions to depend materially on permitted support
  examples and labels under predeclared counterfactual controls.
- A transfer improvement must hold across multiple sealed real datasets and context sizes
  under a matched compute budget. This single-seed protocol does not establish seed robustness.
- Final evaluation must pretrain the candidate afresh, report negative and failed datasets or seeds, and
compare
  against the required matched baselines.
- Publication requires successful sealed final evaluation and evidence that the substantive
```

```
  declared GPU workload ran.
- Scope conclusions to the pinned TabICLv2 configuration, synthetic prior, datasets, context sizes,
  seeds, and compute budget; do not imply reproduction of the released full-scale checkpoint or
  broad tabular generality.

# Manual Task Protocol

This directory is manually authored. Prepare its pinned resources and validate it as described
in README.md; task drafting is not part of its launch path.

The starter uses the pinned v2 classifier stage-one recipe: graph-SCM generation, full model,
1,024 synthetic rows, global batch 64, and four H100s using data parallelism. The immutable
reference is prepared separately on four H100s and cached for each search/final seed. Candidate
and reference receive the same root seeds, 1,765 optimizer steps, synthetic task/example counts,
precision, and 7,200-second training ceiling. Two-GPU runs are timing controls only. When candidates
edit the prior, synthetic values may differ; matching means identical budgets and random seed
schedule, not an immutable prior. The calibrated step count is frozen before optimization.

Search uses training seed 17 and four frozen real tables, each at support sizes 64 and 256
with 128 disjoint query rows. Final evaluation uses fresh seed 101 and four different tables.
One pretrained candidate state serves every table for a seed; no per-table parameter updates are permitted.
Search and final reference caches are separate frozen resources, never visible to candidate code.
Missing, stale, corrupt, or incomplete caches stop evaluation; they never trigger hidden retraining.
The primary score is the equally weighted mean improvement in balanced accuracy over the pinned
baseline. Logistic regression, histogram gradient boosting, and class-frequency comparisons
are reported separately. The evaluator also reports accuracy, log loss, 10-bin calibration error,
durations and training/resource diagnostics. The retained `seed_score_std` field is zero for
one seed and does not measure robustness across seeds.

Both learned models must beat their support-label permutation control by more than 0.01 mean
balanced accuracy per seed. Resetting model state must change predictions (mean absolute
probability change greater than 1e-6) on every case. Restoring the trained state must restore
predictions. Invalid outputs or incomplete training invalidate the whole evaluation with score
-2; all failed cases remain in the evidence inventory. Reduced smoke runs are diagnostics only.

The RPC exposes reset, train, predict (trained or initial state), and status operations. Candidate
code owns model/prior/training policy. The evaluator owns real-table preprocessing and query
labels, split selection, comparisons, scoring, and behavioral controls. Parameter/update counters
and checkpoint digests returned by candidate code are unverified declarations: behavioral controls
alone cannot prove that arbitrary editable Python performed pretraining. Scientific pretraining
and fixed-state claims additionally require source review of the actual selected implementation;
never present those counters as host-attested compute evidence.

Scientific acceptance uses evaluator-observed predictive quality, behavioral controls, host timing,
and the trusted reference's training checks. Candidate-internal counters, state digests, round-trip
flags, and shape hashes remain bounded diagnostics and never determine scientific correctness.
Reference preparation verifies the pinned reference source and records raw training receipts
and all prediction interventions before publishing a cache. Cache identity includes reference
source, runtime image, dependencies, workload, data splits, and control implementation. Candidate
edits do not alter that identity. Cached reference preparation time is reported separately from
current evaluation time. Training children use a fixed Python hash seed so
the unmodified upstream graph prior's set iteration remains reproducible across processes.
```

G.2. Details of Doc-to-LoRA Inductive Bias

Table 11 | Doc-to-LoRA multi-needle NIAH agent-search and evaluation protocol.

Stage	Setting	Value
Agent search	Idea budget	40 ideation cycles
	Model / harness	`gpt-5.6-sol` / Codex; `claude-opus-5` / Claude; `deepseek-v4-pro` / Pi
	Hardware	One H100 GPU with 16 CPU cores
	Training time limit	3 hours; candidates above 3 hours are invalid
Dataset	Training datasets	NIAH samples range from 32 to 256 tokens
	Evaluation datasets	NIAH samples range from 32 tokens to 64K tokens
	Example contract	Each context contains 1-4 needles
	Benchmark scale	512K training samples; 128 samples per length bucket in {128, 8K, 16K, 32K, 64K}, totaling 640 samples.
Baseline model / training	Base model	Fixed `google/gemma-2-2b-it` target model and tokenizer
	Adapter setup	`down_proj` target module and LoRA rank 4

Initial Task Instructions for Doc-to-LoRA Inductive Bias Topic Direction

```
# Goal

Improve the Doc-to-LoRA / Context-to-LoRA hypernetwork on the synthetic
multi-needle NIAH task while keeping training under the strict 3-hour limit on a
single `aisci` GPU. The 3-hour limit is final: runs that require more time are
discarded, not rescheduled or judged separately.

The fixed candidate entrypoint is `program_<idx>/src/program.py`. It is
read-only benchmark code. It already:

1. links the frozen source `ctx_multi_needle_*` jsonl cache from
   `TASK_SOURCE_DIR` when available, then generates missing data only as a
   standalone fallback,
2. runs `scripts/niah/1-train.sh`,
3. evaluates the resulting checkpoint with `scripts/niah/2-eval-test.sh`, and
4. writes `program_<idx>/src/_results.json`.

`evaluate.py` reads `_results.json`, enforces the strict three-hour training limit,
and scores candidates mainly by exact-match accuracy on queried target needles.
Each context independently samples 1--4 named needles, but the prompt asks for
only one target needle so the label remains a single 4-digit number. Candidate
changes should improve the model/training implementation used by that fixed
entrypoint.

# Dataset contract

- Dataset names use the `ctx_multi_needle_<low>_<high>` family. They are
  generated by `program_0/src/data/generate_ctx_multi_needle.py`.
- Each example independently samples `num_needles` uniformly from 1 through 4,
  inclusive. Do not assume every context has the same number of needles.
- Each needle is a named fact such as
  `The needle named amber stores magic number 1234.` Names and 4-digit numbers
  are unique within a context.
- The prompt asks for one target needle by name, for example
  `What magic number is stored by the needle named amber? Reply with only the number.`
  The response is only the 4-digit number for that target needle.
- Target needle depth is balanced across ten context-depth bins over each full
  train/validation/test split, including long token bins with many possible
  context lengths. Distractor needles are placed at random non-target positions,
  so candidates should learn key-value binding rather than copying the first or
  last number.
- This benchmark is not an all-needles recall task. Keep the single-number
  answer contract unless intentionally changing the evaluator and scoring.

# Required background

Read the local paper draft first: `doc_to_lora_paper_icml2026/main.tex` and the
figures/tables it references. It explains the intended Doc-to-LoRA mechanism,
why context-conditioned LoRAs are useful, and which ablations already exist.
```

```
Use it to avoid rediscovering baseline ideas or optimizing the wrong bottleneck.

# Required research directions

Focus edits on real model changes, not evaluator bypasses.

1. **How to merge per-chunk LoRAs.** The current implementation in
   `src/ctx_to_lora/modeling/lora_merger.py` directly concatenates the generated
   A and B matrices from different context chunks. Consider alternatives such as
   normalization, learned or heuristic chunk weighting, low-rank recompression,
   rank budgeting, gating, or merge rules that reduce destructive interference.

2. **Inductive bias in the hypernetwork output space.** Consider where LoRAs are
   applied and what the hypernetwork emits. Examples: which target modules are
   useful (`down_proj`, attention projections, MLP projections), whether all
   layers are necessary, whether rank should vary by layer/module, whether a
   shared basis plus generated coefficients is enough, or whether cheaper output
   spaces such as diagonal/residual scales, layernorm shifts, KV-focused adapters,
   or generated biases give better accuracy per compute.

3. **Inductive bias in the hypernetwork architecture.** The current cross-attention
   design assumes each latent query corresponds to one generated LoRA rank.
   Consider whether that rank-query binding is necessary. Examples: shared latent
   factors with learned rank mixing, hierarchical queries for layer/module groups,
   separate queries for A and B factors, rank tokens conditioned on context
   length, or architectures that generate a compact basis plus coefficients
   before expanding to LoRA parameters.

# Runtime rules

- Training must complete in less than 3 hours on one `aisci` GPU. This limit is
  strict and final. Any run with `train_runtime_seconds > 10800` is invalid and
  will be discarded, regardless of accuracy or partial progress.
- The fixed entrypoint must continue to run the two task scripts named above.
  Do not replace them with fake metrics, cached answers, or a no-op.
- The fixed entrypoint records a `full_epoch_probe` in `_results.json` after
  training. Leave this probe intact; it reports whether full 1-epoch training is
  feasible for the candidate under the strict three-hour limit.
- Default evaluation is intentionally small for iteration, but its environment
  values are fixed by the evaluator. `evaluate.py` strips ambient
  `DOC_TO_LORA_*` variables and injects the benchmark values itself, so every
  candidate sees the same train/eval scale:
  - `DOC_TO_LORA_BASE_SAMPLES_PER_BIN=256000`
  - `DOC_TO_LORA_MAX_TRAIN_SAMPLES_PER_DS=256000`
  - `DOC_TO_LORA_MAX_TEST_SAMPLES_PER_DS=128`
  - `DOC_TO_LORA_EVAL_BATCH_SIZE_GEN=8`
  - `DOC_TO_LORA_MAX_NEW_TOKENS=8`
  - `DOC_TO_LORA_EVAL_DATASETS="ctx_multi_needle_32_128 ctx_multi_needle_7168_8192
  ctx_multi_needle_15360_16384 ctx_multi_needle_28672_32768 ctx_multi_needle_57344_65536"`
- Do not tune scores by changing `DOC_TO_LORA_*` environment-variable usages,
  `evaluate.py`, `program.py`, the data generator, `run_eval.py`, evaluator helpers,
  dataset definitions, or the NIAH shell-script defaults. `program_spec.toml`
  declares these as guarded checks, and the harness rejects candidates that edit
  or bypass them.

# Fixed benchmark boundaries

Candidates should improve model and training internals, not evaluator or runtime
configuration escape hatches.

- Do not edit `evaluate.py`.
- Do not edit `program_<idx>/src/program.py` or change the entrypoint away from
  it.
- Always use the fixed base model `google/gemma-2-2b-it`; do not change
  `model_name_or_path` or add alternate target-model selection.
- Do not add, remove, or modify `DOC_TO_LORA_*` environment-variable usages.
- Do not change the NIAH shell scripts in `scripts/niah/`; they are guarded so
  `program.py` continues to run the same train/eval contract.

# Metrics

The fixed entrypoint writes `_results.json` with at least:

- `training_completed: true`
- `evaluation_completed: true`
- `final_accuracy` in `[0, 1]`
- `exact_match_mean` when generated text files are available
- `train_runtime_seconds`
- `eval_runtime_seconds`
- `checkpoint_path`
- `full_epoch_probe`

`combined_score` is exactly `final_accuracy` for valid runs. Invalid runs receive
a large negative score. `_results.json` and evaluator `metrics.json` also include
```

```
non-scoring diagnostics grouped by needle count, target depth bin, target needle
ordinal, and first/last-target flags.

# Rubrics

- `niah_accuracy`: maximize exact-match accuracy on the generated multi-needle NIAH test set.
- `train_time`: training must stay below 3 hours on one GPU. This is a hard
  cutoff; over-limit runs are discarded.
- `script_contract`: `program.py` must run `scripts/niah/1-train.sh` and
  `scripts/niah/2-eval-test.sh` and write `_results.json`.
- `feasibility_probe`: preserve the fixed entrypoint probe that reports whether
  full 1-epoch training is feasible for the candidate under the time limit.
- `research_focus`: changes should address LoRA merging or hypernetwork output
  inductive bias rather than evaluator shortcuts.
```

G.3. Details of Learned Covariance Matrix Adaptation

Table 12 | MetaEvo training, search-validation, and transfer protocol. BBOB tasks meta-train neural weights; Ant subsequently supplies the score used by the agentic program search.

Stage	Setting	Value
Agent search	Idea iterations	30 per agent
	Models / agents	`gpt-5.6-sol` / Codex; `claude-opus-5` / Claude; `deepseek-v4-flash` / Pi; `kimi-k3` / OpenCode
	Valid evaluations	28 Codex; 30 Claude; 28 DeepSeek + Pi; 30 Kimi + OpenCode
Outer meta-evolution	Optimizer	Full CMA-ES
	Population / generations	256 / 1,500
	Selected candidates' initial std.	0.10 (Codex, Claude, Kimi); 0.01 (DeepSeek + Pi)
BBOB inner loop	Tasks per outer generation	128 newly sampled, shared across candidates
	Population / generations	16 / 50
	Dimension / initial std.	Uniform 2–10 / 1.0
	Functions	Sphere, Rosenbrock, Discus, Rastrigin, Schwefel, Bueche–Rastrigin, Attractive Sector, Weierstrass, Schaffers F7, Griewank–Rosenbrock
	Noise	Additive, standard deviation uniform in $[0, 0.1]$
	Aggregate loss	Per-task standardized loss; median across tasks
Ant search-validation	Population / generations	256 / 2,000
	Seeds / episode length	8 fixed seeds / 500 steps
	Initial standard deviation	0.05
	Score	AUC normalized to initial policy and CMA-ES
Brax transfer	Environments / seeds	8 / 16 new shared seeds each
	Population / generations	256 / 2,000
	Initial std. / episode length	0.05 / 500 steps

Initial Task Instructions for Learned Covariance Matrix Adaptation Topic Direction

````
# Goal

Improve the mutable program under `program_N/src/`. The fixed evaluator meta-trains its
learned parameters only on cheap analytic BBOB functions. It then freezes those parameters,
rebuilds the same optimizer for a Brax Ant policy, and uses the Ant result as the scientific
score. Ant never updates neural weights or outer-CMA state.

This follows the meta-black-box loops used for
[Learned Evolution Strategies](https://arxiv.org/abs/2211.11260) and
[Learned Genetic Algorithms](https://arxiv.org/abs/2304.03995), while testing whether a cheap
analytic training distribution can discover a LearnedCMA that transfers to control. Promising
architectures include permutation-equivariant or self-attention modules that jointly update the
mean and a diagonal, low-rank, factorized, or full covariance representation. Choose the design
from experimental evidence.

# Rubrics

- Maximize the primary CMA-anchored Ant-2000 AUC score; initial policy = 0, CMA-ES = 1.
- Implement a valid evosax-style learned distribution strategy with 1-1,024 meta-parameters.
- Meta-train only on the fixed BBOB distribution, then transfer frozen learned weights to Ant.
- Preserve dimension/population generality and horizon-free, evidence-conditioned updates.
- Keep the fixed evaluator, seeds, budgets, scoring, and runtime contracts unchanged.

# Scientific contract

## Mutable contract

`program.py` must export:

```python
build_candidate(solution, population_size) -> CandidateDefinition
```

`program_0` is a self-contained port of the evosax LearnedES architecture and lifecycle. Its
142 learned weights are freshly initialized and become the initial mean of outer CMA-ES; no
pretrained checkpoint is loaded. You may rewrite its state, features, architecture, sampling,
mean/covariance updates, and the outer-CMA scalar settings exposed by
`CandidateDefinition.outer_settings`.

The learned parameter PyTree must:

- match `strategy.default_params.params` in structure, shape, and dtype;
- contain 1-1,024 finite floating-point scalars;
- work with evosax-style `init`, `ask`, `tell`, and `get_mean` signatures;
- have the same structure, shapes, dtype, initialization, name, strategy type, and outer settings
  when built for 10-D/population-16 BBOB and 704-D/population-256 Ant;
- express domain- and population-size generalization through generic normalized formulas, never
  by branching on the known BBOB/Ant dimensions, population sizes, task identity, or environment.

Every learned distribution decision--including mean, scale, covariance, step-size, learning-rate,
and gating updates--must define an **anytime optimizer**. It must not condition optimizer decisions
on absolute generation, elapsed iteration, the fixed evaluation budget or horizon, a countdown,
or a timestamp embedding. It may condition on horizon-free optimization evidence such as ranks,
population statistics, evolution paths, distribution scale/shape, improvement, success, and
stagnation. A generation counter may exist only for lifecycle bookkeeping and metrics.

## Fixed BBOB meta-evolution

- Outer optimizer: full CMA-ES, population 256, 1,500 generations; final distribution mean selected.
- Inner optimizer: candidate strategy, population 16, 50 generations, initial standard deviation `1.0`.
- Per outer generation: 128 newly sampled tasks, shared across all 256 candidates as common random
  numbers and vectorized over 8 H100 GPUs.
- Training functions: Sphere, Rosenbrock, Discus, Rastrigin, Schwefel, Bueche-Rastrigin,
  Attractive Sector, Weierstrass, Schaffers F7, and Griewank-Rosenbrock.
- Task distribution: dimensions sampled inclusively from 2-10, shifted optima/objective offsets,
  BBOB rotations, and additive Gaussian noise with standard deviation sampled uniformly from
  `[0, 0.1]`.
- Inner diagnostic curve: clean objective regret to the known optimum, transformed with
  `log10(max(regret, 1e-8))`.
- Task loss: best observed objective over all 50 generations and 16 members. Each task is
  standardized across the outer population, then candidate loss is the median across tasks.
  Invalid task runs receive a fixed worst loss.
- Outer settings candidates may alter: `std_init`, `std_min`, `std_max`, `c_mean`, `c_std`,
  `d_std`, `c_c`, `c_1`, `c_mu`, `mu_eff`, `chi_n`, `elite_ratio`, and
  `use_negative_weights`.

The evaluator fixes all task sampling, budgets, hardware, seeds, and scoring. Candidate code may
not access Ant, evaluator reference data, result directories, environment variables, the network,
subprocesses, or files outside its own source bundle. This is a cooperative research-code
contract, not an adversarial sandbox.

## Frozen Ant validation and score
````

````
After BBOB meta-evolution, the selected learned-weight PyTree is serialized, authenticated, and
loaded unchanged into the candidate rebuilt for Ant. No further meta-optimization occurs.

- Environment: historical Brax v1 `ant`, legacy-spring dynamics with contact-force observations.
- Policy: observation-normalized tanh-linear policy, 87 inputs × 8 actions plus bias
  (704 parameters), zero initial search mean.
- Candidate and reference population: 256; initial policy standard deviation `0.05`.
- Evaluation: 8 fixed seeds excluded from BBOB training, one seed per H100, 2,000 generations,
  500-step episodes, one rollout.
- Primary score: full 2,000-generation best-so-far AUC normalized to the pinned same-seed CMA-ES
  reference:

```text
(candidate_AUC_2000 - initial_policy_AUC) / (CMA_ES_AUC_2000 - initial_policy_AUC)
```

Initial policy = 0, CMA-ES = 1, and scores above 1 beat CMA-ES. The evaluator also reports the
first-1,000-generation AUC and best return, generation-2,000 best return, every seed curve, and a
paired bootstrap interval. Those are diagnostics; only the 2,000-generation normalized score
ranks ideas.

Non-finite returns, absolute returns above 10,000, and active-step torso x velocities above
50 m/s are invalid. Observation normalization uses masked population-wide Welford statistics;
invalid trajectories and post-termination AutoReset samples do not update those statistics.

## Operational notes

- Exactly one harness evaluation is allowed per idea. Inspect `metrics.json`, `correct.json`, and
  captured logs before proposing the next architecture.
- BBOB and Ant execute in separate Python processes. BBOB uses normal GPU command buffers; only
  legacy-spring Ant applies the H100 command-buffer workaround.
- The evaluator watchdog is 6h05m inside a six-hour scientific budget plus queue cleanup margin.
- Outer checkpoints bind config, program, evaluator, dependency, and reference identities and the
  serialized state digest. Interrupted runs may resume only through the supplied campaign tools.
- Optimize expected transfer and evidence quality, not evaluator plumbing or benchmark identity.
````

G.4. Details of Error-Diffusion experiments

Table 13 | Error-Diffusion task and evaluation parameters.

Setting	MNIST	CIFAR-10	MuJoCo / Brax	Craftax
Model / Harness	Sonnet 4.6 / Claude Code	Opus 4.6 / Claude Code	Composer 2.5 Fast / Cursor	GPT-5.5 / Codex
Agent search budget	20 ideation cycles	60 ideation cycles	60 ideation cycles	30 ideation cycles
Evaluator resources	1 GPU, 4 CPUs	1 GPU, 8 CPUs	1 GPU, 4 CPUs	1 GPU, 4 CPUs
Dataset / environment	MNIST: 55k train / 5k validation / 10k test	CIFAR-10: 45k train / 5k validation / 10k test	Ant, HalfCheetah, Humanoid	Craftax-Symbolic-v1
Final training budget	250 epochs	250 epochs	400M requested steps per seed	1B requested steps per seed

.

Initial Task Instructions for Error-Diffusion Classification Topic Direction

```
## Goal
- Extend the binary classification Error Diffusion program to a multi-class classification problem. Achieve
at least 95% `val_accuracy` on the fixed MNIST validation split, and aim for more than 98% `val_accuracy` if
possible.
- Do not use the official MNIST test split during search-time evaluation and do not write `test_accuracy`
into `_results.json`.
- Tip: Check gradient norm / activation norm at each epoch during training.

## Rubrics

- Do not use backprop or autograd for the backward pass.
- Do not use W.T (forward weights transposed) in the backward pass, because this is a sign of backprop.
- Make sure to use the MNIST data loader provided in `example_mnist_mlp.py`, using its fixed `55k train / 5k
val / 10k test` split with `split_seed=0` and `500` validation examples per class.
- The training script must write `_results.json` with at least `val_accuracy`, `training_time`,
`terminated_early`, `dataset_name`, `split_seed`, `train_examples`, `val_examples`, and
`val_examples_per_class`.
- `dataset_name` must be exactly `"MNIST"`. Synthetic or toy datasets must not pass evaluation as MNIST.
- Downloaded datasets and caches must not be stored inside any `program_*` directory. Use a shared cache
path such as `AGENTIC_DATA_DIR` when available instead of `./data` relative to the program workspace.
- Keep the core weight updating mechanism as delta w \propto -(learning rate) x (presynaptic activity) x
(postsynaptic activation derivative) x (one global error sign)
- Polarity invariance must be preserved through multiple layers. For this, the following must hold:
  - all trainable weight tensors should remain non-negative (this constraint may be relaxed if needed for
  performance). Any violations of the non-negative constraint should be logged (e.g., the number of violated
  weights).
  - the derivative of the activation function must be non-negative.
  - the forward pass must preserve the excitatory/inhibitory sign structure such as:
    - `p_i = f(+p_{i-1} @ W_pp - n_{i-1} @ W_np + b_p)`
    - `n_i = f(+n_{i-1} @ W_nn - p_{i-1} @ W_pn + b_n)`
- Do not use fixed random matrices or trainable backward weights to route output errors to hidden layers,
because that would make the method equivalent to Direct Feedback Alignment / Feedback Alignment.
- Do not use any data augmentation.
- Do not rely on random feature extractors.
- Do not treat multi-class classification as a simple ensemble of binary ED models.
- Focus on fully connected layer variants; do not explore conv nets or attention.
- Achieve at least 95% validation accuracy on the fixed MNIST validation split.
```

Initial Task Instructions for Error-Diffusion Convolution Topic Direction

```
# Goal
- Extend the best multi-class Error Diffusion program from `ed_mnist` (provided as
`program_0/src/program.py`) to a convolutional neural network. Achieve at least 60% `val_accuracy`
(equivalent to `combined_score`) on the fixed CIFAR-10 validation split, and aim for more than 85%
`val_accuracy` if possible.
- Do not use the official CIFAR-10 test split and do not write `test_accuracy` into `_results.json`.
- Tip: Check gradient norm / activation norm at each epoch during training.

# Rubrics
- Do not use backprop or autograd for the backward pass.
- Do not use W.T (forward weights transposed) in the backward pass, because
  this is a sign of backprop.
```

```
- Make sure to use the same network structure as in `example_cifar10_convnet.py`, but implement it using
Error Diffusion's dual-stream architecture. The model must use convolutional layers. Do not collapse the
task back into a flat MLP on handcrafted features.
- Make sure to use the CIFAR data loader provided in `example_cifar10_convnet.py`, using its fixed `45k
train / 5k val / 10k test` split with `split_seed=0` and `500` validation examples per class.
- The training script must write `_results.json` with at least `val_accuracy`, `training_time`,
`terminated_early`, `dataset_name`, `num_conv_layers`, `split_seed`, `train_examples`, `val_examples`, and
`val_examples_per_class`.
- Print per-epoch training progress including the epoch number and validation accuracy in a parseable format
such as `epoch 012 ... val_acc=0.2345` or `epoch 012 ... val=0.2345`. The evaluator may stop runs that are
learning too slowly based on these logs.
- Optimize against validation accuracy.
- `dataset_name` must be exactly `"CIFAR10"`.
- Downloaded datasets and caches must not be stored inside any `program_*` directory. Use a shared cache
path such as `AGENTIC_DATA_DIR` when available instead of `./data` relative to the program workspace.
- Polarity invariance must be preserved through multiple layers. For this, the
  following must hold:
  - all trainable weight tensors should remain non-negative (this can be relaxed if needed to achieve the
  desired performance).
  - the derivative of the activation function must be non-negative.
  - the forward pass must preserve the excitatory/inhibitory sign structure,
    including convolutional layers, for example:
    - `p_i = f(conv_op(p_{i-1}, W_pp) - conv_op(n_{i-1}, W_np) + b_p)`
    - `n_i = f(conv_op(n_{i-1}, W_nn) - conv_op(p_{i-1}, W_pn) + b_n)`
- Do not use fixed random matrices or trainable backward weights to route output errors to hidden layers,
because that would make the method equivalent to Direct Feedback Alignment / Feedback Alignment.
- Achieve at least 60% validation accuracy on the fixed CIFAR10 validation split.
```

G.5. Details of RL-based LLM fine-tuning

Table 14 | GRPO agent-search and evaluation protocol. A fixed Qwen2.5-Math-1.5B model is trained on the MATH dataset with GRPO; final validation accuracy (mean reward) supplies the score used by the agentic parallel investigation phase, and the best program of each run is re-scored on a held-out test set.

Stage	Setting	Value
Agent search	Idea budget	50 ideation cycles
	Model / harness	`claude-opus-4-7` / Claude
	Program evaluations	1 per idea
	Pruning	Every 20 cycles
	Hardware	Two H100 GPUs with 16 CPU cores
	Training time limit	2.5 hours
	Execution	Containerized (Docker, CUDA 12.8), Slurm-submitted
Dataset	Base model / tokenizer	Fixed `Qwen/Qwen2.5-Math-1.5B`
	Task	MATH reasoning
	Train / validation split	6,000 / 1,500 examples (stratified by subject and level)
	Test split	MATH-500 problems (secret, outside the task mount)

Initial Task Instructions for RL-based LLM fine-tuning Topic Direction

```
# Goal
- Improve Group Relative Policy Optimization (GRPO) for training a language model on math reasoning tasks.
- The base model is Qwen2.5-Math-1.5B, trained on the MATH dataset using GRPO.
- Your goal is to maximize the final validation accuracy (mean reward) of the model.
- Evaluate candidate programs via `evaluate.py` and maximize `combined_score` from `metrics.json`.
- Respect correctness: `correct.json` must report `correct: true`.
- Focus on implementing ONE specific variant of your idea with ONE fixed set of hyperparameters. Do NOT
perform hyperparameter tuning, grid searches, ablation studies, multiple variants, comparative experiments,
or additional baseline comparisons.
- IMPORTANT: Each evaluation runs a full GRPO training loop (20 epochs) and takes approximately 1-2 hours.
When calling `evaluate.py`, you MUST set a bash timeout of at least 7200000 ms (2 hours) to avoid the process
being killed prematurely.

# Rubrics
- `accuracy_target`: final validation accuracy should improve over the baseline.
- `training_stability`: training should complete without NaN losses, OOM errors, or crashes.
- `code_safety`: program must remain executable with the provided evaluator and `grpo_utils` library.
- `evidence`: every evaluated candidate must produce `metrics.json`, `correct.json`, and rubric evidence.
```