WHAT IS WORTH TRAINING NEXT? EVALUATION-GUIDED EVOLUTION FOR AUTONOMOUS TRAINERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous training systems can continually adapt training strategies and improve models with limited human intervention. However, their next optimization directions are often chosen based on limited evaluation feedback such as benchmark scores, which can lead to repeated local optimization. More importantly, existing approaches often assess the value of an optimization direction only after completing an expensive training run. They offer little guidance on **what is worth training next** before the budget is committed. Such guidance is essential when training resources are limited. We introduce **EvalEvo**, an evaluation-guided framework that turns evaluation evidence into prospective training decisions. Before each costly run, EvalEvo acquires targeted evidence to verify observed progress, diagnose the factors limiting further improvement, and assess which candidate directions can provide useful learning signals. It organizes this evidence in a global directed acyclic graph (DAG), allowing future decisions to draw on accumulated cross-version experience. At each iteration, EvalEvo uses cross-version evidence and the current exploration state to select the next training direction and formulate a falsifiable expectation for its outcome. When the existing evaluation or decision process becomes insufficient, EvalEvo also adapts the training system itself to support continued optimization. Across language-model pretraining optimization and reinforcement learning (RL) for mathematical reasoning and code generation, EvalEvo consistently outperforms existing baselines under fixed experimental budgets.

1 INTRODUCTION

Large language models (LLMs) can increasingly participate in their own improvement by proposing training changes, executing them, and learning from previous outcomes (Lu et al., 2024; Yamada et al., 2025; Karpathy, 2026; Jeddi et al., 2026; Qu & Lu, 2026; Chen et al., 2026). This growing autonomy expands the space of training strategies that can be explored with limited human intervention. However, each training run remains expensive, while plausible optimization directions multiply over successive iterations. Under a fixed experimental budget, an autonomous trainer must therefore decide which directions deserve its limited training opportunities. This leads to a central question: *what is worth training next*?

Existing autonomous training systems explore candidate interventions through broad experimentation and guided search, as illustrated in Figure 1. *Broad exploration* trains diverse candidates to cover the search space, while *guided search* uses previous outcomes to prioritize promising directions (Jeddi et al., 2026; Lyu et al., 2026; Ning et al., 2026; Qu & Lu, 2026; Chen et al., 2026). Yet an intervention's past performance does not establish its value for the next training run. A previously successful direction may offer little benefit once its target bottleneck is resolved, while an unsuccessful direction may become useful under different training conditions. Benchmark scores alone cannot distinguish these cases. Deciding what to train next therefore requires connecting observed outcomes to their underlying bottlenecks, available learning signals, and the conditions under which earlier interventions succeeded or failed. This motivates evaluation that actively informs training decisions through targeted investigation and accumulated cross-version evidence.

We introduce **EvalEvo**, an evaluation-guided framework that turns accumulated experimental evidence into subsequent training decisions. Before each full training run, EvalEvo acquires targeted

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

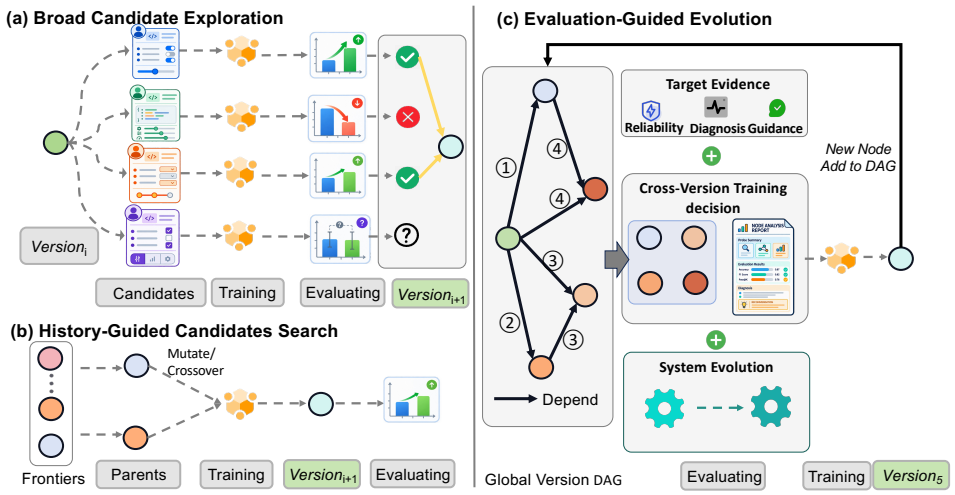


Figure 1: **Evaluation-guided evolution for autonomous training.** (a) Broad exploration trains diverse candidates. (b) Guided search prioritizes candidates using historical outcomes. (c) EvalEvo combines targeted evidence acquisition, cross-version training decisions, and system evolution to determine what is worth training next.

evidence to verify observed progress, diagnose remaining bottlenecks, and identify actionable learning signals. A global version directed acyclic graph (DAG) records this evidence together with interventions and verified outcomes, allowing later decisions to combine findings from multiple versions and revisit earlier conclusions when training conditions change. EvalEvo uses this cross-version evidence to select the next intervention and formulate a falsifiable expectation for its outcome. When existing evaluation tools or decision procedures become insufficient, it also evolves the surrounding training system to support further optimization. We evaluate EvalEvo on language-model pretraining optimization and reinforcement learning for mathematical reasoning and code generation, observing improvements over the compared baselines under matched training budgets.

Our contributions are threefold: (i) we formulate autonomous training around *what is worth training next*, using evaluation evidence to guide training decisions under a limited budget; (ii) we introduce EvalEvo, an evaluation-guided autonomous training framework that evolves training strategies through targeted evidence acquisition and cross-version training decisions over a global version DAG, while evolving the training system itself to support continued optimization; (iii) we demonstrate improvements over the compared baselines in language-model pretraining and RL for mathematical reasoning and code generation under matched training budgets, supported by ablations and trajectory analyses.

2 Related Work

2.1 Self-Evolving Agent Systems

Self-evolving agents seek to improve future behavior by reusing information obtained from prior interactions. Early work mainly uses lightweight mechanisms such as reflection, memory, and reusable skills to convert interaction experience into guidance that can be reused across future tasks (Shinn et al., 2023; Zhao et al., 2024; Ouyang et al., 2025; Wang et al., 2024). More recent methods broaden the targets of self-evolution to system-level components such as tools, workflows, and agent harnesses (Qiu et al., 2025; Zheng et al., 2025a; Hu et al., 2025; Zhang et al., 2025). Self-evolution has therefore expanded from experience retention to system-level adaptation. Recent work on agent harnesses follows this direction, showing that the system around the model can itself become an object of evolution (Zhang et al., 2026; Du et al., 2026; Wei et al., 2026). EvalEvo extends this idea to a costly and resource-intensive setting, where the object of evolution is the training process that updates model parameters. Changes to prompts, skills, or workflows can often be evaluated through additional inference. Weight-level changes are different because their effect is typically

known only after a new training run has been completed. This makes training decisions substantially more expensive to verify. The central question is therefore which training direction is worth allocating compute to next, making budget-aware decision making essential.

2.2 Autonomous Research and Training Optimization

Autonomous research systems increasingly automate the exploration of training and optimization choices. Early systems such as The AI Scientist and AI Scientist-v2 cover broad research workflows, from hypothesis generation to experiment execution and analysis (Lu et al., 2024; Yamada et al., 2025). More recent work focuses more directly on how candidate training directions are generated and organized. Bilevel AutoResearch adapts the search process itself (Qu & Lu, 2026), specialist-agent methods decompose the optimization space across specialized roles (Ning et al., 2026), GEAR maintains and evolves multiple research states (Jeddi et al., 2026), and EvoTrainer combines policy-version exploration with an evolving diagnostic harness (Chen et al., 2026). These methods improve the breadth and structure of autonomous search over candidate experiments. EvalEvo focuses on a complementary decision point. Existing systems mainly use outcomes from completed runs to guide subsequent search, whereas EvalEvo asks whether a direction is worth the next costly training run before that budget is committed. It addresses this problem through prospective evidence acquisition and cross-version reasoning, allowing accumulated training experience to guide which direction should be pursued next under a fixed experimental budget.

3 EvalEvo: Evaluation-Guided Evolution

3.1 Budgeted Autonomous Training

We consider autonomous training under a fixed training budget. At iteration t, a full training run incurs cost c_t, and the total budget is B. Let $S(v)$ denote the headline performance of model version v, and let $\mathcal{V}_T^{\mathrm{ver}}$ be the set of versions verified after T training runs. Our objective is

$$\max_{v \in \mathcal{V}_T^{\mathrm{ver}}} S(v) \qquad \text{s.t.} \qquad \sum_{t=1}^{T} c_t \leq B. \tag{1}$$

The objective is to maximize the performance of the best verified model under a fixed training budget. Since each full training run consumes this budget, the problem naturally reduces to deciding *what is worth training next*. Budget accounting details are provided in Appendix A.3.

3.2 EvalEvo Overview

As illustrated in Figure 2, given a task specification, available resources and tools, and an initial workspace, EvalEvo operates under a fixed training budget through three components. **Targeted Evidence Acquisition** constructs a structured report of reliable observations, diagnosed limitations, and training guidance. **Cross-Version Training Decision** combines this report with historical evidence to determine what is worth training next, producing a concrete intervention and a falsifiable hypothesis. The observed outcome is checked against the hypothesis and recorded in the global version DAG. **Coupled Model-System Evolution** adapts the surrounding training system when its capabilities become insufficient for further optimization.

3.3 Targeted Evidence Acquisition

A benchmark score alone neither establishes the reliability of an observed change nor identifies the remaining limitation. EvalEvo therefore treats evaluation as targeted evidence acquisition. Before the next full training run, the Evaluator probes the current model and constructs

$$R_t = \left(E_t^{\mathrm{rel}}, E_t^{\mathrm{diag}}, E_t^{\mathrm{pre}} \right), \tag{2}$$

where R_t is the structured evidence report at iteration t, and E_t^{rel}, E_t^{diag}, and E_t^{pre} denote reliability, diagnostic, and prescriptive evidence, respectively, for the subsequent training decision.

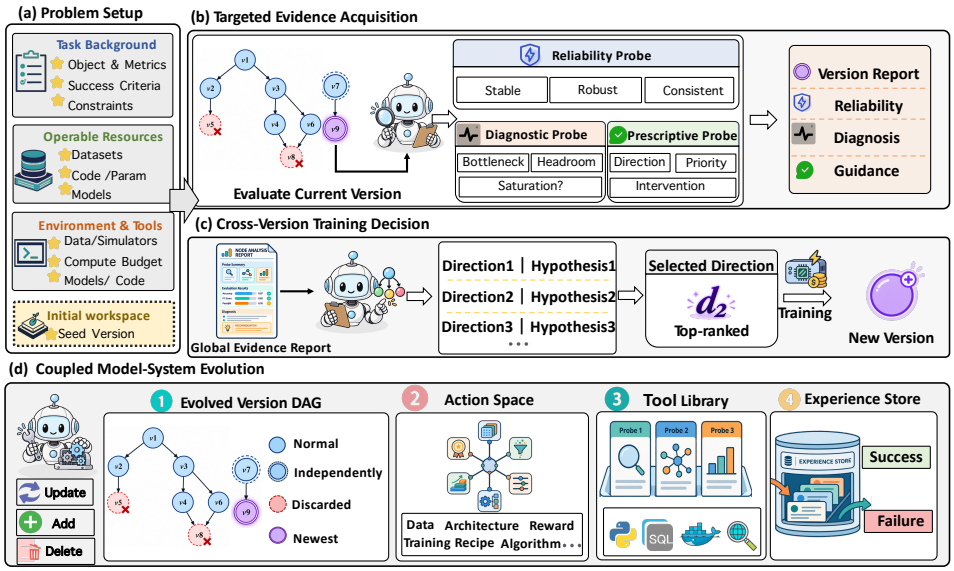


Figure 2: **Overview of EvalEvo.** Under a fixed training budget, EvalEvo proceeds through four stages: (a) problem setup defines the task and available training context; (b) targeted evidence acquisition characterizes the current model; (c) cross-version training decision-making selects the next optimization direction using current and historical evidence; and (d) coupled model-system evolution adapts the training system to sustain further improvement.

Reliability evidence determines whether an observed change should be treated as a real training signal rather than a noisy outcome. Instead of directly interpreting a higher benchmark score as improvement, EvalEvo checks whether the change persists under alternative evaluation conditions. The Evaluator can resample examples, construct targeted subsets, inspect prediction consistency, compare neighboring model versions, or rerun controlled evaluations with fixed decoding conditions. These checks separate reproducible capability changes from effects caused by sampling variance, narrow-set overfitting, or incomplete optimization. For each completed training run, EvalEvo checks the observed outcome against the hypothesis recorded in its training decision (Section 3.4). A trial is marked positive when the evidence supports the hypothesis and confirms a reliable improvement in the fixed headline metric. The verification result is stored with the corresponding version, including failed or inconclusive outcomes, to inform subsequent training decisions.

Diagnostic evidence asks a different question: even if the observed result is reliable, what is preventing the model from improving further? Aggregate scores provide little information about where the remaining headroom lies. EvalEvo therefore decomposes the observed behavior through controlled comparisons and targeted probes. The Evaluator may contrast successful and failed examples, construct queries that isolate specific capabilities, vary one factor while holding others fixed, inspect rollout trajectories, or perform lightweight short-run interventions when observational evidence is insufficient. These probes are used to distinguish between limitations arising from data coverage, reward specification, optimization dynamics, generation behavior, or other task-specific factors. The output is not merely an error category, but an estimate of which bottleneck currently dominates and whether removing it is likely to expose additional headroom.

Prescriptive evidence identifies which parts of the diagnosed bottleneck are actionable through further training. Rather than selecting a concrete intervention, EvalEvo looks for usable learning signals in the current model behavior, such as recurring failure patterns, separable reward or validation signals, and informative rollout statistics. Diagnostic evidence therefore identifies *where* improvement is constrained, while prescriptive evidence reveals *what can be acted upon*. Evidence is acquired through an extensible evaluation harness that combines data construction, model inference, programmatic analysis, external search, and lightweight experimental tools. These capabilities support targeted probes for the uncertainty exposed by each model version. The resulting evidence

report is then stored with the corresponding model version and used by the cross-version decision module in Section 3.4 to determine what is worth training next.

3.4 Cross-Version Training Decision

The evidence in Section 3.3 characterizes the current model, while accumulated experimental history provides additional context for deciding what to train next. EvalEvo combines current evidence with experimental history to reason over an extensible action space $\mathcal{A}_t$, whose initial scope is specified in Appendices A.1 and A.4. The history is organized as a global version DAG,

$$\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t), \tag{3}$$

where each node stores a model version together with its evidence, intervention, hypothesis, and outcome. A directed edge records the direct reuse of findings or interventions from an earlier version in constructing a later training decision. These edges capture specific dependencies, while the broader experimental history remains available for candidate scoring.

Using the current evidence and $\mathcal{G}_t$, EvalEvo constructs a set of candidate optimization directions,

$$\mathcal{D}_t = \left\{ d_t^{(1)}, \ldots, d_t^{(K)} \right\}, \tag{4}$$

where each $d_t^{(k)}$ specifies an aspect of the training process to modify. To balance evidence support, historical success, and exploration, EvalEvo ranks each direction using an equally weighted score,

$$J_t(d) = C_t(d) + S_t(d) + U_t(d), \tag{5}$$

where $C_t(d) \in [0, 1]$ is the trainer's confidence in the direction, based on evaluation evidence, prior knowledge, and accumulated experience. The historical success and exploration terms are

$$S_t(d) = \frac{n_t^+(d) + 1}{n_t(d) + 2}, \qquad U_t(d) = \frac{1}{\sqrt{n_t(d) + 1}}, \tag{6}$$

with $n_t(d)$ counting relevant previous trials and $n_t^+(d)$ counting those with verified positive outcomes. Together, these terms favor supported directions while retaining opportunities to explore those with limited trial history. The highest-ranked direction is instantiated as a concrete training decision,

$$T_t = (\mathcal{P}_t, \Delta_t, H_t), \qquad \mathcal{P}_t \subseteq \mathcal{V}_t, \tag{7}$$

where $\mathcal{P}_t$ identifies the earlier versions directly used to construct the intervention Δ_t, and H_t states a falsifiable hypothesis about its expected effect. The dependency set can include multiple versions or be empty, allowing new directions without direct dependencies on existing nodes. In our RL experiments, every candidate is initialized from the same base model, so these dependencies transfer experimental knowledge and recipe components rather than model weights. After training, the outcome is checked against H_t and recorded with the new version in the DAG, with incoming edges from $\mathcal{P}_t$. Because dependencies always point from earlier to later versions, the graph remains acyclic but need not be connected.

3.5 Coupled Model-System Evolution

EvalEvo couples model evolution with evolution of the surrounding training system. While model updates produce new versions and evidence, system evolution changes the context and capabilities available for subsequent training decisions. We represent this evolvable system state as

$$\Phi_t = (\mathcal{G}_t, \mathcal{A}_t, \mathcal{U}_t, \mathcal{M}_t), \tag{8}$$

where $\mathcal{G}_t$ is the version DAG, $\mathcal{A}_t$ the action space, $\mathcal{U}_t$ the tool library, and $\mathcal{M}_t$ the experience store.

These components capture complementary aspects of the training system. The version DAG maintains evolving cross-version evidence dependencies; the action space defines the training modifications currently available; the tool library provides evaluation and analysis capabilities; and the experience store preserves reusable knowledge distilled from previous trials. EvalEvo can selectively update these components when the current system becomes insufficient for subsequent optimization. The optimization objective and headline evaluation remain fixed throughout this process. Thus, system evolution changes how EvalEvo reasons about and executes future training decisions, without changing what constitutes model improvement.

4 RESULTS

4.1 AUTORESEARCH: EFFICIENT AUTONOMOUS EXPERIMENTATION

Table 1: AutoResearch results under a common budget of 100 training experiments. "First exp. to beat AutoResearch" is the earliest experiment whose validation bpb is lower than the final AutoResearch result. VRAM is the peak memory consumption of the reported configuration.

Variant	bpb ↓	VRAM (GB) ↓	Params (M)	First exp. to beat AutoResearch ↓
AutoResearch (Karpathy, 2026)	0.9980	62.57	83.9	–
GEAR-EVOLVE (Jeddi et al., 2026)	0.9935	43.90	50.3	69
EvalEvo (ours)	**0.9899**	**33.17**	94.4	**23**

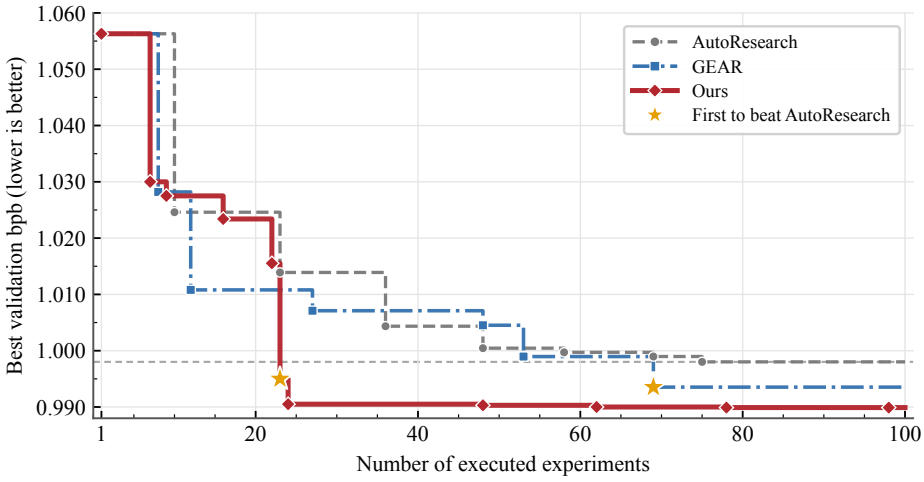


Figure 3: **AutoResearch optimization trajectories.** Running-best validation bpb over executed training experiments under the same 100-run budget. The horizontal reference marks the final AutoResearch result, and the star denotes the first EvalEvo run that surpasses it.

Tasks and Protocol. We evaluate EvalEvo on AutoResearch (Karpathy, 2026), where an autonomous agent iteratively modifies a language-model training program and evaluates each proposal through an actual training run. The trainer may modify only `train.py`, while the data and evaluation utilities in `prepare.py` remain fixed. We follow the original task and evaluation protocol and use validation bits per byte (bpb) as the scalar optimization objective, with lower values indicating lower normalized negative log-likelihood on the fixed validation data. The validation data correspond to the fixed held-out validation shard defined by the original AutoResearch data pipeline and are excluded from training. All methods start from the same unmodified training program and receive the same budget of 100 experiments, with each training run capped at 300 seconds. We compare EvalEvo with the original AutoResearch loop and GEAR-EVOLVE (Jeddi et al., 2026), representing single-incumbent and population-based autonomous search, respectively. Details are provided in Appendix A.1.

Performance under a Fixed Budget. We use validation bits per byte (bpb) as the scalar objective for AutoResearch and optimize for lower bpb throughout the search. EvalEvo achieves both faster progress and a stronger final solution under the same experimental budget. It reaches a validation bpb of **0.9899**, outperforming GEAR-EVOLVE (0.9935) and AutoResearch (0.9980) in Table 1. More importantly, EvalEvo surpasses the final AutoResearch result after only **23 experiments**, compared with 69 for GEAR-EVOLVE. Figure 3 shows that the gap opens well before the budget is exhausted and continues to widen thereafter. The final configuration also uses only 33.17 GB of peak VRAM, versus 43.90 GB for GEAR-EVOLVE and 62.57 GB for AutoResearch, indicating that the improvement is not obtained by uniformly increasing resource consumption.

Why Does EvalEvo Perform Better? The AutoResearch trajectory shows that the value of an intervention depends on the training regime in which it is tested. Earlier width expansion provides little benefit, so model capacity initially appears close to saturation. At $v22$–$v23$, full-context attention substantially improves throughput, increasing MFU from 21.1% to 38.9% and changing the effective optimization regime. EvalEvo then revisits the previously weak width direction at $v24$. Increasing width from 640 to 768 improves bpb from 0.9950 to 0.9905, even though the wider model completes fewer training steps than its parent. This sequence illustrates a key advantage of cross-version reasoning: EvalEvo does not permanently classify a direction as useful or exhausted from a single earlier result. Instead, it re-evaluates previous conclusions when later evidence changes the conditions under which they were formed. Detailed verification and prospective decisions are provided in Appendix B.

4.2 Autonomous RL Optimization on Math and Coding

Tasks and Protocol. Following EvoTrainer (Chen et al., 2026), we study RL for mathematical reasoning and code generation. Math uses BigMath-Hard (Albalak et al., 2025) for training and AIME 2024/2025 (Mathematical Association of America, 2026) and CNMO 2024 (Chinese Mathematical Society, 2024) for evaluation; Coding uses cleaned TACO (Li et al., 2023) and LiveCodeBench-v6 (Jain et al., 2025), respectively. All methods share the data and evaluation protocol, with every training run initialized from the same Qwen3.5-9B base model (Qwen Team, 2026). We report Avg@8 over eight rollouts per item with seed 42 (Appendix A.2). EvalEvo initially supports modifying training hyperparameters, reward functions, RL algorithms, and recipes. Its operations, diagnostic tools, and decision procedures can evolve, while the headline evaluation protocol stays fixed. Appendix A.4 specifies the initial scope and evaluation-feedback access.

Baselines and Training Budget. We compare three baseline groups: *reference configurations*, comprising the base model and a human-engineered RL recipe; *algorithmic baselines*, comprising Group-Variance Filtering (Yu et al., 2025), GRPO (Shao et al., 2024) with Clip-Higher (Yu et al., 2025), and GSPO (Zheng et al., 2025b) with Clip-Higher, covering rollout filtering, policy optimization, and clipping; and *autonomous training*, represented by AutoResearch (Karpathy, 2026), which iteratively searches training configurations. Within each domain, autonomous methods share the initialization and a budget of eight completed weight-updating runs, each capped at 3,200 optimizer steps on 32 NVIDIA H20 GPUs, as detailed in A.3.

Compute Accounting. EvalEvo proactively acquires targeted evidence and performs adaptive evaluation before allocating subsequent training runs, introducing decision-support computation beyond weight-updating RL. We explicitly account for this cost rather than treating it as part of the matched training budget. Up to the point of first surpassing the human-engineered reference, EvalEvo uses 17.92 GPU-hours of decision-support computation on Math and 27.36 GPU-hours on Coding, corresponding to only 0.40% and 0.44% of the realized training compute, respectively. This overhead is small relative to the training cost and does not materially change the compute comparison. Detailed accounting is provided in Appendix A.3.

Table 2: Math and Coding results (Avg@8, %). EvalEvo and AutoResearch report their best results within eight training rounds. Numbers in parentheses denote absolute gains over the base model.

	Math				Coding
Method	**AIME24**	**AIME25**	**CNMO24**	**Avg**	**LCB-v6**
Reference Configurations					
Base Model	77.92	62.50	75.00	71.81	47.21
Human-engineered RL	80.83	63.75	81.25	75.28	51.29
Algorithmic Baselines					
Group-Variance Filtering	78.75	60.42	76.39	71.85	50.71
GSPO+Clip-Higher	80.42	63.33	78.47	74.07	50.29
GRPO+Clip-Higher	80.83	63.33	79.17	74.44	50.93
Autonomous Trainers					
AutoResearch (Karpathy, 2026)	78.33	63.33	78.47	73.38	51.86
EvalEvo	**82.92** (+5.00)	**64.58** (+2.08)	**82.64** (+7.64)	**76.71** (+4.90)	**53.50** (+6.29)

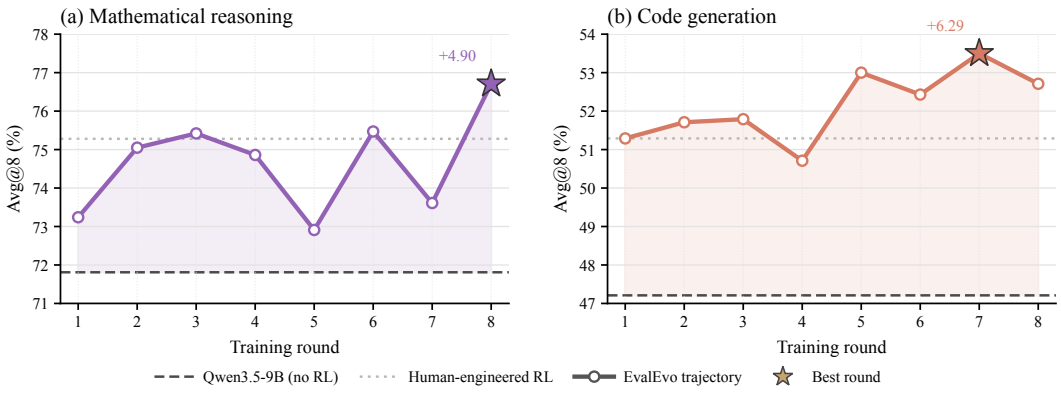


Figure 4: **RL evolution trajectories on Math and Coding.** Performance after each chronological training round under the same eight-run budget.

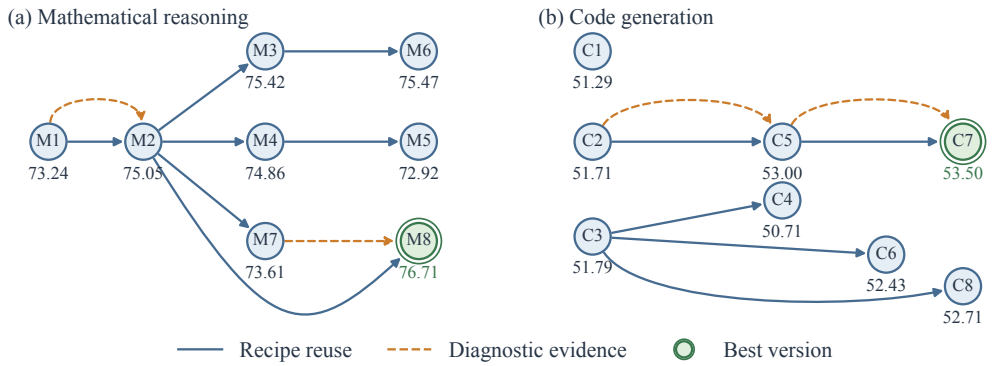


Figure 5: RL evolution trajectories. Nodes show version IDs and Avg@8 scores (%). Solid and dashed arrows indicate recipe and diagnostic evidence reuse, respectively; neither denotes model-weight inheritance. Green double outlines mark the best versions.

Performance under a Matched RL Training Budget. Table 2 shows that EvalEvo achieves the best result on every benchmark within the eight-run budget and is the only autonomous method to surpass the human-engineered recipe in both domains. On Math, it improves Avg@8 from 71.81 to 76.71, with gains of +5.00 on AIME 2024, +2.08 on AIME 2025, and +7.64 on CNMO 2024, exceeding the human recipe by 1.43 points on average. On Coding, it raises LiveCodeBench-v6 from 47.21 to 53.50, outperforming the human recipe and AutoResearch by 2.21 and 1.64 points, respectively. Figure 4 shows an overall upward trend with local regressions. EvalEvo already surpasses the human reference at rounds 3 and 2 on Math and Coding, yet continued evolution raises the best results until rounds 8 and 7, respectively.

Figure 5 shows how these trajectories emerge from evolving diagnoses and cross-version decisions; M_i and C_i denote the Math and Coding versions at round i. For example, analysis of M_1 reveals that many long failed responses have already reached the correct answer but continue generating until truncation. This distinguishes a misassigned learning signal from insufficient reasoning and motivates the reward correction in M_2. A later diagnostic branch, M_7, identifies a remaining inconsistency in how such trajectories are rewarded; M_8 incorporates this evidence into the stronger M_2 recipe. In Coding, C_2 prioritizes difficult but learnable problems at the cost of broader training coverage. C_5 restores balanced coverage and simultaneously adds a reward intended to suppress repetitive endings, while C_7 removes this additional reward to test whether it is necessary. These transitions illustrate how EvalEvo resolves competing explanations, reuses evidence across branches, and separates bundled changes before allocating another training run. They culminate in the selected best versions M_8 and C_7, which achieve 76.71 and 53.50 Avg@8 and improve over their recipe sources M_2 and C_5 by 1.66 and 0.50 points, respectively. Complete traces and averaged Pass@8 curves appear in Appendices C and D, respectively.

4.3 Ablation Study

We isolate targeted evidence acquisition and system evolution through two ablations. *w/o Probe* removes targeted probing while retaining standard score-based evaluation and the remaining decision process. *w/o System Evolution* keeps the initial evidence-acquisition procedures and training-decision tools fixed, preventing them from incorporating newly developed procedures or distilled experience across runs. The version DAG remains available and continues to record interventions, evidence, and outcomes; this ablation therefore tests whether accumulated history is sufficient without adapting how that history is acquired and used. We do not construct a separate *w/o Cross-Version* ablation, since this reduces the decision process to a sequential, history-local search setting already represented by AutoResearch (Karpathy, 2026). All variants share the same starting model and training budget, with details in Appendix A.5.

Table 3: Ablations on AutoResearch under the same training budget. "First beat" denotes the first experiment with lower bpb than the final AutoResearch baseline; – indicates that this threshold is not reached.

Variant	bpb ↓	VRAM (GB) ↓	Params (M)	First beat ↓
EvalEvo	**0.9899**	**33.17**	94.4	**23**
w/o Probe	0.9934	43.90	50.3	37
w/o System Evolution	1.0300	43.90	50.3	–

Table 4: Controlled ablations on mathematical reasoning and code generation. All variants share the same starting model and training-attempt budget. Scores are Avg@8 percentages.

Variant	AIME24	AIME25	CNMO24	Math Avg	LCB-v6
EvalEvo	**82.92**	**64.58**	**82.64**	**76.71**	**53.50**
w/o Probe	79.58	63.75	81.25	74.86	50.71
w/o System Evolution	78.33	63.33	79.86	73.84	50.79

Tables 3 and 4 reveal different roles for the two components. On AutoResearch, removing targeted probing degrades bpb from 0.9899 to 0.9934 and delays the first improvement over the AutoResearch baseline from experiment 23 to 37. The ablated system can still discover a strong configuration, but requires 14 additional training attempts, indicating that targeted evidence primarily improves which uncertainties are resolved before the next expensive run. Without system evolution, the best bpb falls to 1.0300 and never surpasses the baseline. Because the DAG remains available in this variant, this failure shows that recording more experiments alone is insufficient when fixed evaluation and decision procedures can no longer extract useful next steps from them.

The RL ablations show the same distinction: removing targeted probing lowers Math and Coding Avg@8 by 1.85 and 2.79 points, while fixing the training system lowers them by 2.87 and 2.71 points. Targeted probing improves decisions within the current system, whereas system evolution adapts the evaluation and decision processes as new failure modes emerge. Their complementary effects support both components of the full framework.

5 Conclusion

We study autonomous training as a finite-budget decision problem centered on what is worth training next. EvalEvo combines targeted evidence acquisition, cross-version reasoning, and system evolution to make each costly training opportunity more effective under a fixed budget. We evaluate this framework in two distinct settings: language-model pretraining optimization with AutoResearch, and reinforcement learning for mathematical reasoning and code generation. Across both settings, EvalEvo reaches stronger models under the same training budget, while ablations confirm the contributions of targeted evidence acquisition and system evolution. These results show that evaluation can serve not only as a measure of progress, but also as a guide for how an autonomous trainer should evolve under a limited training budget. Future work can extend EvalEvo to broader intervention spaces, longer optimization horizons, and settings with multiple objectives or heterogeneous resource constraints.

AI USE STATEMENT

Large language models were used both in conducting the research and in preparing the manuscript. During the research process, they were used as components of the autonomous training system for tasks including evaluation, diagnosis, training-intervention proposal, experiment analysis, and decision support. They were also used for language editing, code assistance, and debugging during manuscript preparation. All research objectives, experimental protocols, reported results, interpretations, and conclusions were defined or verified by the authors. The authors take full responsibility for the content of this work.

ETHICS STATEMENT

This work focuses on methods for autonomous model training and does not involve human-subject studies or the collection of personal data. All experiments are conducted on established research datasets and benchmarks. We are not aware of additional ethical concerns specific to this work beyond those generally associated with large-scale model training and evaluation.

REPRODUCIBILITY STATEMENT

We provide the experimental setup, training and evaluation protocols, training budgets, and implementation details used in the main experiments in Section 4 and Appendix A. The supplementary material contains the code and configurations used for the reported experiments. The complete implementation and supporting materials will be released upon publication.

REFERENCES

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, et al. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*, 2025.

Anthropic. Claude code. `https://docs.anthropic.com/en/docs/claude-code`, 2026. Accessed: 2026-09-26.

Guhong Chen, Yingcheng Shi, Yongbin Li, Binhua Li, Xander Xu, Hu Wei, Shiwen Ni, Min Yang, and Jieping Ye. Evotrainer: Co-evolving llm policies and training harnesses for autonomous agentic reinforcement learning. *arXiv preprint arXiv:2606.03108*, 2026.

Chinese Mathematical Society. Chinese national high school mathematics olympiad, 2024. URL `https://www.cms.org.cn/Home/comp/comp/cid/12.html`.

Yuetian Du, Yucheng Wang, He Xu, Jiexu Xu, Shanwen Tan, Bing Zhao, Boyu Yang, Zhijie Xu, Ming Kong, Hu Wei, Jie Liu, and Qiang Zhu. Living-harness is an interactive-agent evolver. *arXiv preprint arXiv:2607.26598*, 2026.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL `https://openreview.net/forum?id=chfJJYC3iL`.

Ahmadreza Jeddi, Minh Ngoc Le, Hakki C. Karaimer, Konstantinos G. Derpanis, and Babak Taati. GEAR: Genetic autoresearch for agentic code evolution. *arXiv preprint arXiv:2605.13874*, 2026.

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

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

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*, 2024.

Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, Lun Zhou, and Xiaohui Yan. Evoscientist: Towards multi-agent evolving AI scientists for end-to-end scientific discovery. *arXiv preprint arXiv:2603.08127*, 2026.

Mathematical Association of America. American invitational mathematics examination (aime), 2026. URL `https://maa.org/maa-invitational-competitions/`.

Jingjie Ning, Xiaochuan Li, Ji Zeng, Hao Kang, and Chenyan Xiong. Auto research with specialist agents develops effective and non-trivial training recipes. *arXiv preprint arXiv:2605.05724*, 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025. doi: 10.48550/ARXIV.2509.25140. URL `https://doi.org/10.48550/arXiv.2509.25140`.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Yaonan Qu and Meng Lu. Bilevel autoresearch: Meta-autoresearching itself. *arXiv preprint arXiv:2603.23420*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025.

Tianxin Wei, Zhan Shi, Minhua Lin, Bing He, Zewen Liu, Yisi Sang, Yuanchen Bei, Xuying Ning, Jiaru Zou, Ting-Wei Li, et al. Evo-harness: Context-to-harness skill compilation for self-evolving agents. *arXiv preprint arXiv:2608.15071*, 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.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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 Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025a.

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, 2025b. URL https://arxiv.org/abs/2507.18071.

APPENDIX

A EXPERIMENTAL SETUP AND REPRODUCIBILITY

This section provides supplementary details on the experimental environment, data preparation, evaluation protocol, and training-budget accounting. Task definitions, benchmark descriptions, and primary evaluation metrics are provided in the main text.

A.1 AUTORESEARCH SETUP

Agent Configuration. We instantiate both the Evaluator and Trainer using Claude Code with `claude-opus-4-8` at high reasoning effort (Anthropic, 2026). All runs use the same agent instructions, tool permissions, and workspace configuration.

Task and search space. For language-model pretraining optimization, we follow the original AutoResearch setup and codebase. The fixed utilities in `prepare.py`, including data preparation, tokenizer construction, data loading, and evaluation, are kept unchanged. The optimization system is allowed to modify only `train.py`, which contains the model architecture, optimizer, training loop, batch configuration, and other training choices.

Data and evaluation protocol. We follow the original AutoResearch data pipeline and train on the ClimbMix-400B-Shuffle corpus (Karpathy, 2026). Evaluation is performed on the fixed validation data defined by the original setup, which are excluded from training. The context length is fixed at 2,048 tokens, and each evaluation covers $40 \times 524{,}288 = 20{,}971{,}520$ validation tokens. Model quality is measured by validation bits per byte (`val_bpb`), which normalizes token-level negative log-likelihood by the corresponding byte count; lower values indicate better predictive performance. The training data split, validation data, and evaluation procedure are kept identical across all methods and throughout the optimization process.

Hardware and execution environment. All methods, including the reproduced AutoResearch baseline, are run in the same single-GPU environment using NVIDIA H20 GPUs. Each experiment is assigned an exclusive GPU and does not share the device with other training runs, ensuring that differences in optimization outcomes are not caused by hardware contention or heterogeneous execution environments.

A.2 REINFORCEMENT LEARNING SETUP

We study mathematical reasoning and code generation with Qwen3.5-9B (Qwen Team, 2026). All RL training runs are implemented with the ROLL framework (Wang et al., 2025). Each full training run uses 32 NVIDIA H20 GPUs, each with 96 GB of memory. All candidates are initialized from the same base model; EvalEvo evolves the training recipe rather than continuously inheriting model weights from the preceding candidate.

Agent Configuration. We instantiate both the Evaluator and Trainer using Claude Code with `claude-opus-4-8` at high reasoning effort (Anthropic, 2026). All runs use the same agent instructions, tool permissions, and workspace configuration.

Tasks and Data. For Math, we use 6,428 training problems from BigMath-Hard (Albalak et al., 2025). Following the original data construction, 6,429 problems are first retained and one example overlapping with AIME 2024 (Mathematical Association of America, 2026) is removed. Evaluation covers 78 competition problems: 30 from AIME 2024, 30 from AIME 2025 (Mathematical Association of America, 2026), and 18 from CNMO 2024 (Chinese Mathematical Society, 2024). For Coding, we use 9,872 cleaned and instruction-formatted training problems from TACO (Li et al., 2023). Evaluation contains 175 problems from LiveCodeBench-v6 (Jain et al., 2025).

Autonomous-Training Initialization. Following EvoTrainer (Chen et al., 2026), we separate human-provided initialization from subsequent autonomous optimization. Before evolution begins,

humans bootstrap the task, evaluation benchmarks, training data, compute environment, initial training playbooks, and the initial runnable training version. In the RL setting, the initial action space $\mathcal{A}_0$ covers modifications to training hyperparameters, reward functions, RL algorithms, and training recipes. This defines the initial scope of intervention; EvalEvo can extend the action space during optimization as described in Appendix A.4.

For fair comparison, EvalEvo and the corresponding autonomous-training baselines share the same initialization within each domain. Every RL candidate is independently initialized from the same base model, without inheriting weights from previously evaluated candidates. Cross-version reuse transfers evaluation evidence, intervention outcomes, and training-recipe knowledge. After initialization, each method evolves independently under the same training budget using its own evaluation and decision procedure. No additional method-specific training knowledge is introduced at initialization.

We do not include a direct reproduction of EvoTrainer (Chen et al., 2026) because the publicly available materials are insufficient to reconstruct its experimental pipeline under our matched-budget setting. In particular, the paper describes the overall autonomous training loop and reports task-level results, but does not provide a released implementation that specifies the full agent-side execution logic, experiment orchestration, and decision procedure used to produce the reported trajectories. We also could not identify enough information to unambiguously recover the exact base-model initialization associated with each reported experiment, together with the corresponding training configurations and intermediate decision states. These details are necessary for a faithful comparison because autonomous training performance depends not only on the outer-loop algorithm, but also on the initial checkpoint, accessible tools, execution policy, stopping criteria, and how observations from previous runs are translated into subsequent interventions. Re-implementing these unspecified components would therefore introduce substantial researcher-dependent choices and risk comparing against our own reconstruction rather than the reported EvoTrainer system. We consequently cite EvoTrainer as closely related work but exclude it from the reproduced baseline table.

A.3 Training Budget and Compute Accounting

We distinguish the *training budget*, which constrains the available weight-updating training opportunities, from the *realized compute cost*, which additionally accounts for computation used to support training decisions.

AutoResearch. For AutoResearch, we follow the benchmark protocol: each method receives 100 training experiments, with each experiment capped at 300 seconds of weight-updating training on a single NVIDIA H20 GPU. Since the training program itself can change, the number of optimizer steps and processed tokens may vary across experiments, while neither the number of experiments nor the per-experiment wall-clock limit can be exceeded. EvalEvo introduces no additional GPU-based decision-support computation beyond the standard experimental loop in this setting.

Math and Coding RL. For Math and Coding, each autonomous trajectory is limited to 8 completed weight-updating runs, with every run capped at 3,200 optimizer steps on the same 32 NVIDIA H20 GPUs. A run may terminate early, but unused steps cannot be converted into additional training runs. Because rollout lengths and early termination can lead to different realized costs even under this matched training budget, we additionally record GPU-hours for each trajectory.

We separate the realized cost into *training GPU-hours*, covering rollout generation, forward and backward computation, and optimizer updates within each RL run, and *decision-support GPU-hours*, covering inference-only targeted evidence acquisition, diagnosis, verification, and adaptive evaluation between runs. Any operation that updates model parameters is counted as training compute. We report the relative decision-support overhead as

$$\rho_{\mathrm{dec}} = \frac{\text{Decision-Support GPU-h}}{\text{Training GPU-h}} \times 100\%. \tag{9}$$

To compare compute efficiency at a common performance level, we measure the cumulative GPU-hours required to first surpass the human-engineered RL reference, rather than the cost of reaching each method's own best checkpoint. On Math, the reference achieves 75.28 Avg@8. EvalEvo

surpasses it at round 3, while AutoResearch does not reach this level within the eight-run budget. On Coding, the reference achieves 51.29 Avg@8; EvalEvo reaches it at round 2, compared with round 3 for AutoResearch.

Table 5: Compute required to first surpass the human-engineered RL reference on Math (75.28 Avg@8). Costs are cumulative through the first round exceeding the reference.

Method	Round	Training GPU-h	Decision-Support GPU-h	Decision / Training (%)
AutoResearch	Not reached	–	–	–
EvalEvo	3	4512	17.92	0.40

Table 6: Compute required to first surpass the human-engineered RL reference on Coding (51.29 Avg@8). Costs are cumulative through the first round exceeding the reference.

Method	Round	Training GPU-h	Decision-Support GPU-h	Decision / Training (%)
AutoResearch	3	7104	0	0
EvalEvo	2	6240	27.36	0.44

The additional decision-support cost is small relative to training compute: 17.92 GPU-hours (0.40%) on Math and 27.36 GPU-hours (0.44%) on Coding at the point where EvalEvo first surpasses the human-engineered reference. On Coding, including this overhead, EvalEvo reaches the reference with 6267.36 GPU-hours, compared with 7104 GPU-hours for AutoResearch; on Math, EvalEvo reaches the reference within three rounds whereas AutoResearch does not reach it within all eight.

A.4 Trainer Scope and Feedback Access

Trainer Scope. EvalEvo operates over an extensible intervention space. The trainer can propose and execute changes that it judges promising for improving model performance, subject to the prescribed training budget and fixed evaluation protocol. Its scope includes modifying training methods, processing or generating training data, conducting inference experiments, and developing new diagnostic tools and decision procedures. Table 7 provides representative operations rather than an exhaustive list; the trainer can extend these operations as new evidence and optimization needs emerge.

Table 7: Representative operations available to EvalEvo in the RL experiments. The intervention space can expand during optimization.

Area	Example operations
Training methods	Modify training hyperparameters, reward functions, RL algorithms, and training recipes, and execute candidate training runs.
Training data	Inspect, filter, transform, reweight, or generate training data, and revise data mixtures based on evaluation evidence.
Inference experiments	Compare models, generation settings, and prompting strategies; generate and analyze rollouts to investigate failures and assess candidate training directions.
Training system	Develop or revise tools, diagnostic procedures, and decision processes, and maintain cross-version evidence for subsequent optimization.

Feedback Access. The trainer can inspect training logs, benchmark scores, and associated evaluation rollouts to guide interventions and model selection. The headline benchmark instances, scoring rules, and evaluation settings remain fixed throughout optimization, while training-side diagnostics and experimental procedures can evolve. Each candidate training run starts from the same base model; cross-version reuse transfers evidence and training recipes without inheriting candidate weights. Reported best scores reflect performance attained under this benchmark-feedback protocol.

A.5 Ablation Protocols

The ablations use the same task-specific initialization, training-budget limits, and headline evaluation protocol as the full framework. They differ in targeted evidence acquisition and adaptive updates to the training system, as specified below.

Without Targeted Probing. This variant retains standard score-based evaluation but disables targeted evidence acquisition. The remaining training-decision process and system-evolution mechanisms are retained.

Without System Evolution. This variant retains targeted evidence acquisition using the initial procedures, but disables subsequent changes to these procedures and to training-decision tools. It also freezes the experience store $\mathcal{M}_t$ at its initial state, preventing the addition of reusable experience distilled from successful or failed runs. The version DAG $\mathcal{G}_t$ continues to accumulate version records, evaluation evidence, interventions, and outcomes.

This distinction separates experimental records in $\mathcal{G}_t$ from reusable experience in $\mathcal{M}_t$. Records in $\mathcal{G}_t$ remain available for retrieval during subsequent decision making, while $\mathcal{M}_t$ no longer accumulates reusable experience from prior runs. The ablation therefore evaluates the combined effect of adaptive procedures, tools, and experience accumulation while retaining access to the growing experimental record.

B Detailed AutoResearch Evolution Trace

This section provides process-level evidence for two behaviors that are central to EvalEvo in the AutoResearch setting: evolving the evaluation process when the current signal becomes insufficient for reliable decisions, and revisiting previously weak interventions when the surrounding training regime changes.

System Evolution When Evaluation Becomes the Bottleneck. Early in the trajectory, a single validation score is sufficient to distinguish clearly better training configurations. Near the performance frontier, however, improvements become small enough that score differences are comparable to evaluation noise. At this stage, the bottleneck is no longer only which intervention to try, but whether the observed gain provides enough evidence to justify spending another training run in that direction.

EvalEvo responds by evolving the evaluation procedure itself. It introduces an additional held-out check and a tolerance of $\epsilon = 0.002$ bpb, treating changes within this range as inconclusive. A direction is accepted only when the observed improvement is supported beyond this tolerance by both evaluations. Across the first 24 evolved versions, no accepted direction improves on the primary validation score while failing this additional check. The key effect is therefore not simply a more elaborate evaluator: EvalEvo increases evaluation rigor only when the existing signal becomes insufficient for allocating the remaining training budget reliably.

Cross-Version Reasoning for Regime-Dependent Interventions. The sequence leading to the first configuration that surpasses the AutoResearch baseline illustrates a second behavior: the value of an intervention can change after the training regime itself changes. Table 8 summarizes the relevant versions.

Table 8: Key versions in the AutoResearch breakthrough sequence. Lower validation bpb is better.

Version	Intervention	Val. bpb	Steps
$v22$	More full-context attention	1.0155	1,568
$v23$	All layers full-context	0.9950	2,251
$v24$	Width $640 \rightarrow 768$	0.9905	1,838

Earlier in the trajectory, increasing model width provides little benefit, making additional capacity appear to be a weak direction. This conclusion changes after $v23$. Switching all layers to full-context

attention raises the number of optimization steps completed within the same wall-clock budget from $1{,}568$ to $2{,}251$. EvalEvo therefore revisits model width under this new regime. At $v24$, increasing the width from 640 to 768 reduces validation bpb from 0.9950 to 0.9905.

This gain cannot be attributed simply to more optimization steps, since $v24$ completes only $1{,}838$ steps, fewer than the $2{,}251$ steps of $v23$. Instead, the evidence suggests that the value of additional capacity depends on the surrounding training regime. The earlier negative result is therefore not treated as a permanent rejection of width, but as evidence conditioned on the regime in which it was observed. Cross-version reasoning allows EvalEvo to retain this conditional evidence and revisit the intervention once those conditions change.

C Detailed RL Evolution Trace

This section examines how evaluation evidence informs training decisions throughout the Math and Coding trajectories. Figure 5 summarizes the cross-version dependencies, while the following tables and analyses connect individual interventions to their diagnostic evidence. All candidates are initialized from the same base model; dependencies transfer recipes and evidence rather than model weights.

C.1 Evolution Trace: Mathematical Reasoning

Table 9 summarizes the Math interventions and their dependencies. The trajectory explores context length, reward design, and data coverage, with the final version combining an earlier training decision with diagnostic findings from a separate branch.

Table 9: Math interventions and dependencies. R denotes training decision reuse and E denotes diagnostic evidence reuse. A dash indicates no dependency on an earlier displayed version.

Version	Depends on	Intervention	Decision context
M1	–	Extend context from 4K to 8K.	Examine truncation under longer generations.
M2	M1 (R, E)	Correct truncation rewards and suppress post-answer rambling.	Address reward-label errors exposed in M1.
M3	M2 (R)	Balance data difficulty and strengthen rambling control.	Explore data coverage within the corrected recipe.
M4	M2 (R)	Add a boxed-answer gate.	Test an answer-termination intervention.
M5	M4 (R)	Extend context to 16K and use an asynchronous pipeline.	Test further context extension under the revised pipeline.
M6	M3 (R)	Add clean-easy replay and a boxed-answer gate.	Revisit easy-example coverage within the data branch.
M7	M2 (R)	Investigate reward integrity.	Produce diagnostic findings subsequently used by M8.
M8	M2 (R); M7 (E)	Apply the full reward-integrity correction.	Combine an earlier recipe with findings from another branch.

Diagnosing long-response failures. Early Math versions show a strong concentration of errors among long responses. Responses shorter than 2,000 tokens achieve over 97% correctness, while correctness falls to roughly 69% between 6,000 and 8,000 tokens and to 9% near the generation limit. These aggregate statistics could suggest shortening the response budget. However, trajectory inspection reveals that 219 out of 221 validation responses without a final `\boxed{}` answer reach the 8,192-token limit. Some long trajectories have already produced a correct answer before continuing until truncation, potentially losing useful positive training signal under the original reward treatment.

This evidence motivates correcting how long and truncated trajectories contribute to training, including softer treatment of over-length responses and separate handling of post-answer continuation, while preserving generation capacity for difficult problems. The diagnostic distinction matters: a correlation between response length and failure does not by itself establish whether to shorten generation or correct the reward assigned to the resulting trajectories.

Reusing evidence across versions. M8 combines the training decision from M2 with reward-integrity findings from M7, reaching a Math Avg@8 of 76.71%. Although M7 achieves a lower score of 73.61%, its diagnostic findings remain useful for constructing the subsequent intervention. This dependency distinguishes the value of a version's decision from the value of its evidence: the next decision can reuse an earlier recipe while incorporating findings from another branch.

Refining the evaluation process. Correctness and response length initially reveal the association between long generations and failure, but do not distinguish incorrect reasoning, missing final answers, truncation before completion, and unnecessary continuation after a correct answer. These cases require different training interventions. EvalEvo therefore extends the evaluation process with final-answer presence, truncation status, and post-answer continuation signals. This refinement supports a more specific diagnosis of whether a long response reflects a capability failure or a mismatch between a valid solution and its assigned training signal, informing the subsequent reward correction.

C.2 Evolution Trace: Code Generation

Table 10 summarizes the Coding trajectory. Separate branches examine repetition control, data coverage, and optimization changes. The C2–C5–C7 branch revisits the training-data distribution and then separates the effect of the tail guard from the combined intervention.

Table 10: Coding interventions and dependencies. R denotes training decision reuse and E denotes diagnostic evidence reuse. A dash indicates no dependency on an earlier displayed version.

Version	Depends on	Intervention	Decision context
C1	–	Penalize repeated tails.	Test repetition control as a training direction.
C2	–	Use a reachable-frontier curriculum.	Explore focused sampling and its effect on data coverage.
C3	–	Use sequence-level importance sampling.	Establish an optimization branch for subsequent interventions.
C4	C3 (R)	Apply cross-query batch normalization.	Test a normalization change within the C3 decision.
C5	C2 (R, E)	Balance difficulty coverage and add a tail guard.	Revisit the narrowed coverage observed in C2.
C6	C3 (R)	Filter for informative groups.	Test filtering within the C3 decision.
C7	C5 (R, E)	Retain balanced coverage and remove the tail guard.	Separate the tail guard from the combined changes in C5.
C8	C3 (R)	Use multi-level execution rewards.	Test denser reward feedback within the C3 decision.

Learning from a lower-scoring version. The frontier-focused curriculum in C2 exposes a trade-off between concentrating on difficult but learnable examples and maintaining broader data coverage. The resulting version provides useful diagnostic evidence even though it does not achieve the highest score. C5 revisits this decision by restoring more balanced difficulty coverage and adding a tail guard, improving Coding Avg@8 from 51.71% to 53.00%. The transition illustrates how a subsequent intervention can respond to limitations revealed by an earlier experiment, rather than simply retaining or discarding its decision based on the aggregate score.

Refining a combined intervention. Because C5 changes both data coverage and the tail guard, its aggregate improvement does not isolate their respective contributions. C7 retains the coverage

adjustment and removes the tail guard, reaching 53.50%, the highest observed Coding score. This comparison supports retaining balanced coverage without the additional tail guard in this trajectory and illustrates how a combined intervention can be refined through a subsequent training decision.

The remaining branches also distinguish improvement over a source recipe from improvement over the best available version. C6 and C8 score 52.43% and 52.71%, respectively, both above their recipe source C3 at 51.79%, but neither exceeds C7. The selected best version is therefore C7, rather than the final chronological version C8.

D Additional RL Evaluation

Table 11 reports the complete eight-round evaluation trajectories for Math and Coding, including both Avg@8 and Pass@8. For Math, we report AIME 2024, AIME 2025, and CNMO 2024 separately together with the aggregated Math results. Figure 6 complements the Avg@8 trajectories in the main text with the corresponding Pass@8 evolution.

Table 11: **Complete eight-round RL evaluation trajectories.** Results are reported as percentages. Math Avg. and Math Pass summarize performance across AIME 2024, AIME 2025, and CNMO 2024.

	Math								Coding	
	AIME24		AIME25		CNMO24		Overall		LCB-v6	
Round	Avg@8	Pass@8	Avg@8	Pass@8	Avg@8	Pass@8	Avg@8	Pass@8	Avg@8	Pass@8
1	80.83	93.33	60.42	83.33	78.47	**100.00**	73.24	92.22	51.29	72.00
2	81.67	93.33	62.92	83.33	80.56	**100.00**	75.05	92.22	51.71	72.00
3	79.58	93.33	**65.42**	83.33	81.25	**100.00**	75.42	92.22	51.79	72.00
4	81.67	93.33	63.75	83.33	79.17	94.44	74.86	90.37	50.71	73.14
5	79.58	93.33	60.00	83.33	79.17	94.44	72.92	90.37	53.00	74.86
6	**82.92**	**96.67**	62.92	83.33	80.56	**100.00**	75.47	**93.33**	52.43	71.43
7	79.17	93.33	60.42	83.33	81.25	**100.00**	73.61	92.22	**53.50**	**76.00**
8	**82.92**	93.33	64.58	**86.67**	**82.64**	94.44	**76.71**	91.48	52.71	**76.00**

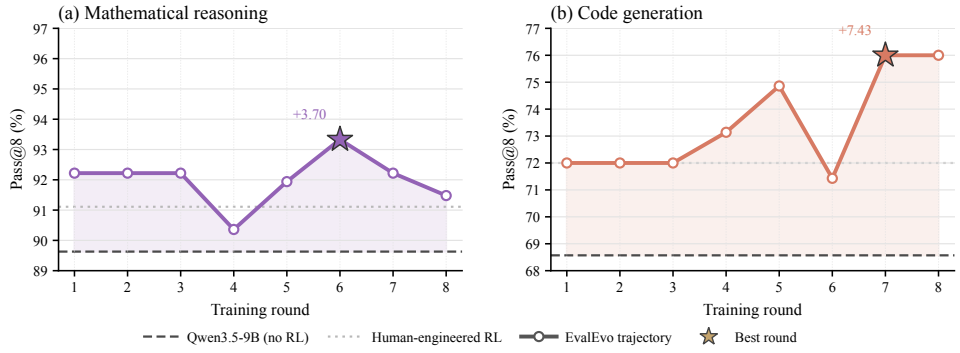


Figure 6: **Pass@8 evolution across the eight EvalEvo training rounds.** (a) Aggregated Pass@8 on the three Math benchmarks. (b) Pass@8 on LiveCodeBench-v6. Pass@8 measures whether at least one of eight generations succeeds on each problem and is reported as a complementary measure to Avg@8.

The round-by-round results show that optimization is non-monotonic and that different Math benchmarks peak at different rounds. Math Avg@8 increases from 73.24 in round 1 to its best value of 76.71 in round 8, while AIME 2025 peaks earlier at round 3 and AIME 2024 reaches its highest value at rounds 6 and 8. In comparison, aggregated Math Pass@8 varies within a narrower range and reaches its highest value of 93.33% at round 6. The larger variation in Avg@8 indicates that the optimization primarily changes how consistently the model succeeds across repeated generations, rather than only whether a problem can be solved at least once.

Coding exhibits a similar distinction. Avg@8 reaches its maximum of 53.50 in round 7, while Pass@8 reaches 76.00% in rounds 7 and 8 after a temporary decline in round 6. We therefore use Avg@8 as the primary metric for comparing round-to-round performance and report Pass@8 as a complementary measure of solution coverage.