EFFECTPRECONCEPTION: GUIDING ML RESEARCH AGENTS THROUGH EFFECT PREDICTION AND OBSERVATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous machine-learning research agents optimize models through iterative code modifications, relying on scalar validation scores to guide exploration. A validation score reveals whether performance changed, but not whether the edit achieved its intended effect, leaving agents unable to separate flawed ideas from uncalibrated implementations. We introduce EffectPreconception, a framework that separates the predicted effect of an edit from its performance outcome. Before execution, the agent states an explicit, testable prediction about the observable behavior the edit should induce. When domain decompositions or accumulated observations support it, an outcome relation maps this predicted effect to an expected change in validation score. After execution, EffectPreconception checks the measured effect against the prediction and records the result in a persistent evidence ledger that informs every subsequent proposal, including evidence from discarded candidates. We evaluate EffectPreconception on Autoresearch, MLE-bench Lite, and PostTrainBench. On Autoresearch, it reduces mean best validation bits per byte from 0.9501 to 0.9404 across three runs with Sonnet 5. Applied to MLEvolve, it increases the mean gold-medal rate on MLE-bench Lite from 56.1% to 65.2% with Opus 5. These results demonstrate that evaluating targeted structural effects alongside validation performance provides substantially better guidance for autonomous research agents than score-only feedback or direct score prediction. Code: `https://anonymous.4open.science/r/EffectPreconception-8022`.

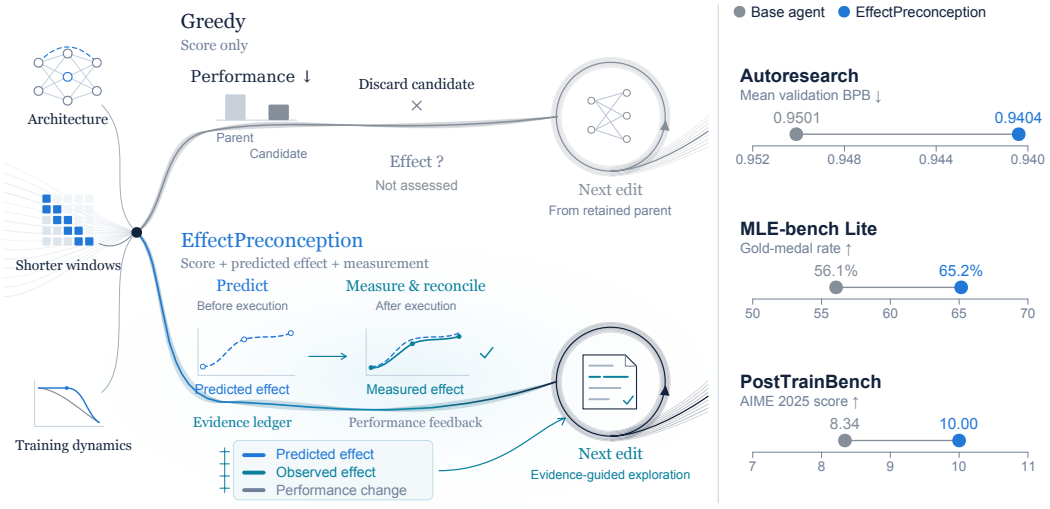


Figure 1: **Guiding research with testable effect predictions.** Greedy search judges edits by validation score, potentially discarding promising ideas without checking their intended effects. EffectPreconception tests effect predictions against measurements and preserves evidence from retained and discarded candidates to guide subsequent proposals. The right panels show mean best validation BPB over three Autoresearch runs using Sonnet 5, mean gold-medal rate over three MLE-bench Lite runs, and mean AIME 2025 score across four PostTrainBench models.

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.

1 INTRODUCTION

Autonomous machine learning (ML) research agents conduct research through successive code changes and experiments, using validation scores to compare candidate models and guide further development (Huang et al., 2024; Jiang et al., 2025; Nam et al., 2025). To guide subsequent proposals, agents can use verbal reflection and textual feedback (Shinn et al., 2023; Yuksekgonul et al., 2025; Agrawal et al., 2026), while persistent memory carries lessons from successful and failed attempts into later tasks (Ouyang et al., 2026). Agents can also draw on past experience to estimate objective values or compare expected performance when choosing which candidates to execute (Liu et al., 2024; Zheng et al., 2026). Once a selected edit has run, its validation score shows whether performance improved, but leaves open whether the edit changed the model or training process as intended. An edit may fail to improve performance because it did not produce the intended effect, or because the effect occurred but did not help. This distinction matters for the next proposal, since revising how an idea is implemented and reconsidering its expected benefit lead to different follow-up experiments.

To make this distinction concrete, we define an edit's effect as a measurable change in the model or training process, assessed through a specified observable and comparison reference. Consider an edit that introduces learnable gates to attention-head outputs (Qiu et al., 2025) to suppress unhelpful contributions. A worse validation score leaves open whether the gates remained near neutral values or changed the outputs without improving performance. In one recorded experiment, measurements showed suppression in the final layer alongside amplification in earlier layers. Given the candidate and these measurements, a revision agent constrained the gates to attenuation only, recovering the lost performance and improving on the ungated parent (Appendix E.1). The discarded candidate thus provided evidence for refining the gating design that its validation score alone did not supply.

We introduce EffectPreconception to make this connection between an edit's expected effect, its measured behavior, and subsequent proposals part of the research cycle (Figure 1). The framework expresses a preconception, an expectation about an edit's effect formed before execution, as a testable effect prediction. Before the experiment runs, it records the expected change or pattern and how it will be measured (Nosek et al., 2018). When a known decomposition or prior observations support it, an outcome relation connects the predicted effect to an expected change in validation performance. After execution, reconciliation assesses the effect prediction against the measured effect without using the proposing agent's rationale or final validation score, and evaluates any supported performance estimate separately. This keeps the effect prediction assessable even when no outcome relation is available. An evidence ledger then preserves the prediction, measurements, and assessments alongside the observed performance outcome and any supported outcome relation. By returning this evidence to the agent before subsequent proposals, the framework allows both retained and discarded candidates to inform further research.

We test whether this additional evidence improves iterative search on language-model training optimization with Autoresearch (Karpathy, 2026), ML engineering tasks on MLE-bench Lite (Chan et al., 2025) using MLEvolve (Du et al., 2026), and language-model post-training on PostTrainBench (Rank et al., 2026). With Sonnet 5 and a 1,200-second training budget per node, EffectPreconception reaches a mean best validation score of 0.9404 bits per byte (BPB) after 100 nodes, compared with 0.9501 for Autoresearch across three runs. On MLE-bench Lite, it raises MLEvolve's mean gold-medal rate from 56.1% to 65.2% with Opus 5 across three runs. To examine the role of predictions within this feedback, our Autoresearch ablation compares the full framework with a version that retains measured history and outcome relations but omits effect predictions. The full framework achieves better search performance, supporting the value of checking prior expectations against experimental observations.

We make the following contributions.

- We formulate experimental feedback around an edit's intended effect and its expected performance benefit. Checking these separately distinguishes an edit that did not behave as intended from one that behaved as intended but failed to improve performance, providing different grounds for the next proposal.
- We introduce EffectPreconception, a framework that turns these expectations into testable predictions for ML research agents. It compares predicted effects with measured effects and, when supported by prior evidence, uses outcome relations to assess the expected

change in validation performance. An evidence ledger carries these findings from both retained and discarded candidates into subsequent proposals.

- We demonstrate that EffectPreconception helps agents discover better models and solutions across Autoresearch, MLE-bench Lite, and PostTrainBench. Autoresearch ablations further show that explicitly predicting effects and checking them against observations improves search beyond retaining measured history and outcome relations alone.

2 Related Work

Autonomous machine-learning research agents. Autonomous ML research agents improve code through repeated experiments, as evaluated by MLAgentBench and MLE-bench (Huang et al., 2024; Chan et al., 2025). The AI Scientist systems extend this process to idea generation and scientific reporting (Lu et al., 2026; Yamada et al., 2025). AIDE, AIRA, MLE-STAR, and MLEvolve organize exploration through search, targeted refinement, and accumulated experience (Jiang et al., 2025; Toledo et al., 2025; Nam et al., 2025; Du et al., 2026). Autoresearch applies this workflow under fixed training budgets (Karpathy, 2026). TextGrad, GEPA, and ReasoningBank use textual feedback, reflection, or stored experience to guide subsequent improvements (Yuksekgonul et al., 2025; Agrawal et al., 2026; Ouyang et al., 2026). Our focus is the evidence returned to the agent after each experiment.

Prediction-guided experiment selection. World models guide decisions by predicting the consequences of actions (Hafner et al., 2025); DSWorld brings this idea to data-science workflows through transition prediction (Yang et al., 2026). Related predictive approaches estimate resource costs or model performance from code and configurations, as in Regression Language Models for Code and LLM-PP (Akhauri et al., 2026; Jawahar et al., 2024). Belief-Calibrated Optimization (BCO) uses predictions to update persistent beliefs: it records behavioral and aggregate expectations before scaffold edits and checks them against execution (Chen et al., 2026). For adaptive experiment selection, Sara combines an LLM agent with a statistical optimization backend (Brunzema et al., 2026). LLAMBO uses LLM surrogates to estimate objective values and uncertainty for Bayesian optimization (Liu et al., 2024). FOREAGENT selects candidate programs through pairwise performance predictions conditioned on code and a dataset report (Zheng et al., 2026). EffectPreconception tests whether an edit produced its predicted observable effect and returns this assessment alongside score feedback to guide subsequent proposals.

3 Method

3.1 Overview

EffectPreconception expresses an agent's preconception as a testable effect prediction and carries the resulting evidence into subsequent proposals (Figure 2). After execution, reconciliation checks the prediction against recorded measurements without using the proposer's rationale or final score, while assessing any performance prediction separately. An evidence ledger preserves these assessments for retained and discarded candidates, leaving parent selection, candidate retention, and final selection to the base policy. The complete procedure is given in Algorithm 1 (Appendix A.2).

Let x_t denote the candidate program or pipeline produced at iteration t and x_p its previously evaluated parent. For each candidate x_i, let r_i denote its recorded measurements and y_i its evaluation score. The agent searches under a fixed research budget, with scores oriented so that higher values indicate better performance. We define the performance change relative to the parent as $\Delta y_t = y_t - y_p$. Let $\mathcal{L}_{t-1}$ denote the evidence ledger available before iteration t. The agent's proposal context h_t combines the parent, experiment history, and a summary of this ledger. The prediction record C_t for the proposed edit contains registered or free-form effect predictions alongside any supported outcome relations and the performance changes they predict.

3.2 Effect predictions and outcome relations

EffectPreconception supports registered effect cards and free-form predictions to describe the measurable changes or patterns expected from an edit. For registered cards, the task implementation specifies observable quantities and measurement rules, including comparison references, units, measurement resolutions, and inputs or observation intervals. Given the edit, its rationale, and the parent context, the agent can specify one or more registered effect cards, each predicting how a selected

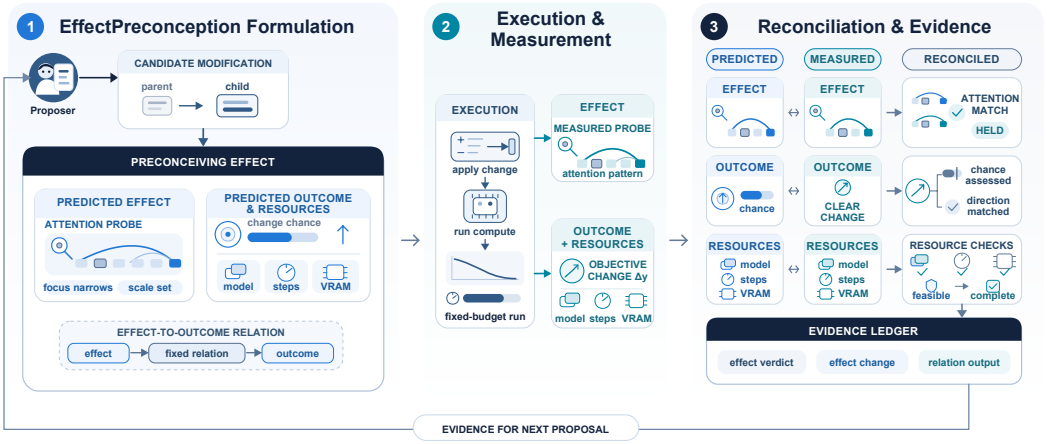


Figure 2: **Overview of EffectPreconception.** Before execution, the agent records an effect prediction alongside auxiliary predictions of performance and resource use (left), with a supported outcome relation mapping the predicted effect to an expected performance change when available. Execution then supplies measurements (middle) for reconciliation to assess the effect prediction independently of the performance outcome. The evidence ledger preserves this assessment alongside the observed performance change to inform the next proposal (right).

observable will change relative to the parent. For a registered effect card in C_t, we write it as

$$c_t = (q_t, d_t, m_t), \tag{1}$$

where q_t identifies the observable quantity and $d_t \in \{-1, 0, +1\}$ denotes the predicted direction, corresponding to a decrease, no discernible change, or an increase, respectively. The optional magnitude m_t is nonnegative when supplied; otherwise, the card specifies only the predicted direction.

Free-form predictions let the agent define the observable and expected pattern for a particular edit. A separate prediction call receives the edit and its rationale, then specifies a measurement-only probe and the pattern expected in its output. The probe collects measurements without modifying the candidate, supporting comparisons with the parent or patterns within a run, such as deviations from neutral values or differences across layers or training phases. To make the prediction specific to the edit, the predictor is asked to choose a measurement that would differ if the intended effect did not occur. Auxiliary predictions and any framework-supplied magnitudes are recorded separately from the agent's effect prediction, with their construction and task-specific measurement rules detailed in Appendix A.

Let M_t denote an outcome relation that maps a predicted or measured effect to an expected performance change. The framework constructs this relation from a known decomposition or fits it to earlier pairs of observed effects and performance changes measured under a common protocol, using the parent context where required. The relation can be used with either prediction form when its input, unit, measurement, and history requirements are met, and is fixed before execution. For a prediction with a supported outcome relation, let $\hat{e}_t$ denote the predicted effect expressed in the relation's input unit. For a registered card with a usable magnitude, this is the signed magnitude $d_t m_t$, after any required unit conversion. The corresponding expected performance change is

$$\widehat{\Delta y}_t = M_t(\hat{e}_t). \tag{2}$$

When no supported outcome relation is available, the framework still compares the predicted effect with the measured effect and records the assessment for subsequent proposals, but does not estimate a score change from that effect.

3.3 Reconciling Predictions with Observations

A better score does not establish that an edit produced its predicted effect, so reconciliation assesses the effect prediction using recorded measurements alone. For a registered card, a mechanical check computes the observed effect, denoted by e_t, from the parent and candidate measurements r_p and

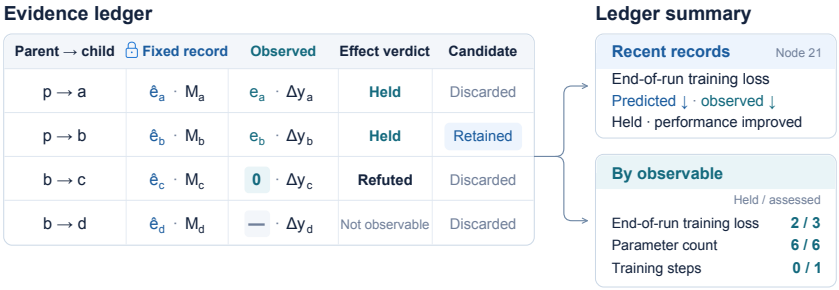


Figure 3: **Evidence ledger and ledger summary.** The schematic ledger links predictions, observations, and assessments for retained and discarded candidates. The summary combines recent records with per-observable statistics for subsequent proposals.

r_t using the card's measurement rule. Applying the measurement resolution, the check returns held if the observed direction agrees with d_t, or refuted otherwise, and records magnitude agreement separately. For free-form predictions, a separate call uses LLM-based assessment (Zheng et al., 2023) to compare the expected pattern with probe outputs without seeing the edit's rationale or final score. We denote the resulting effect assessments for the prediction record C_t by a_t. Unavailable or invalid measurements receive a not observable assessment, distinguishing missing evidence from an observed absence of change.

Performance predictions are assessed separately by comparing the expected change $\widehat{\Delta y}_t$ with the observed change Δy_t. To examine how the effect prediction contributes to this error, reconciliation substitutes the observed effect e_t into the same fixed outcome relation. This substitution requires both predicted and observed effects to satisfy the relation's input requirements, including its declared unit. The resulting estimate $M_t(e_t)$ gives the performance change expected from the measured effect, allowing the signed prediction error to be decomposed as

$$\widehat{\Delta y}_t - \Delta y_t = \underbrace{M_t(\hat{e}_t) - M_t(e_t)}_{\text{effect prediction contribution}} + \underbrace{M_t(e_t) - \Delta y_t}_{\text{outcome relation error}} . \tag{3}$$

The first term measures how much the performance estimate changes when the predicted effect is replaced by the observed effect, while the second is the error remaining after this replacement. Both terms describe prediction error in performance units and supplement the direct assessment of the effect prediction.

Keeping these checks separate gives the agent evidence about which expectation to revisit. A refuted effect prediction directs attention to a mismatch between expected and measured behavior, while the magnitude assessment identifies overestimated or underestimated changes. When the effect occurs as predicted but performance worsens, the agent can instead reconsider the effect's expected benefit. These situations can produce the same score yet suggest different follow-up investigations, a distinction that a direct performance prediction alone cannot provide (Appendix A.3).

3.4 Evidence for subsequent proposals

The evidence ledger $\mathcal{L}_t$ contains the records accumulated through iteration t, including assessed experiments whose candidates were discarded, so their evidence remains available to subsequent proposals. Each entry links the original effect prediction and its source to the parent and edit, preserving the measurements and effect assessment alongside the observed performance change and any outcome relation with its assessment.

Before each proposal, the framework appends a ledger summary to the agent's task, parent context, and experiment history (Figure 3). Recent records let the agent revisit individual predictions alongside their measured effects, assessments, and available performance estimates and outcomes. Statistics grouped by observable or edit family help identify recurring discrepancies, reporting how often predictions held and how predicted magnitudes compare with measurements. For supported outcome relations, the summary also reports the supporting history and errors from earlier predictions fixed before execution. The agent can thus use both recent examples and accumulated evidence when choosing its next edit and predicting its effect.

Table 1: **Autoresearch results.** Validation bits per byte (BPB, lower is better) of the best program retained after the given number of nodes, with 1,200 s of training per node. Each experiment is run independently three times. Recovery is computed within each run as node-1 BPB minus node-100 BPB (higher is better). Bold marks the best mean for each backbone, including ties. All methods run on the Autoresearch harness.

Method	Base LLM	Node 20	Node 40	Node 60	Node 80	Node 100	Recovery ↑
Autoresearch	Sonnet 5	0.9689 ± 0.0005	0.9673 ± 0.0011	0.9510 ± 0.0017	0.9505 ± 0.0020	0.9501 ± 0.0013	0.0250 ± 0.0076
ScorePred	Sonnet 5	0.9499 ± 0.0024	0.9489 ± 0.0016	0.9484 ± 0.0019	0.9480 ± 0.0013	0.9480 ± 0.0013	0.0314 ± 0.0013
LLAMBO	Sonnet 5	0.9560 ± 0.0035	0.9557 ± 0.0034	0.9532 ± 0.0043	0.9518 ± 0.0040	0.9503 ± 0.0040	0.0245 ± 0.0050
FOREAGENT	Sonnet 5	0.9507 ± 0.0040	0.9503 ± 0.0039	0.9492 ± 0.0029	0.9477 ± 0.0025	0.9456 ± 0.0028	0.0339 ± 0.0109
EffectPreconception	Sonnet 5	**0.9478** ± 0.0009	**0.9429** ± 0.0019	**0.9408** ± 0.0017	**0.9406** ± 0.0020	**0.9404** ± 0.0024	**0.0414** ± 0.0060
Autoresearch	Opus 5	0.9622 ± 0.0031	0.9575 ± 0.0022	0.9514 ± 0.0026	0.9496 ± 0.0029	0.9483 ± 0.0007	0.0293 ± 0.0040
ScorePred	Opus 5	0.9499 ± 0.0026	0.9480 ± 0.0016	0.9469 ± 0.0019	0.9458 ± 0.0010	0.9450 ± 0.0013	0.0324 ± 0.0055
LLAMBO	Opus 5	0.9520 ± 0.0028	0.9493 ± 0.0026	0.9480 ± 0.0024	0.9478 ± 0.0026	0.9462 ± 0.0029	0.0244 ± 0.0059
FOREAGENT	Opus 5	0.9545 ± 0.0003	0.9510 ± 0.0009	0.9502 ± 0.0007	0.9487 ± 0.0008	0.9486 ± 0.0009	0.0228 ± 0.0032
EffectPreconception	Opus 5	**0.9477** ± 0.0021	**0.9461** ± 0.0047	**0.9428** ± 0.0061	**0.9381** ± 0.0063	**0.9368** ± 0.0043	**0.0416** ± 0.0017

Completed experiments also supply the history for refining future outcome relations. Valid parent–child pairs enter this history only after their predictions have been assessed, whether the candidate was retained or discarded. New observations can therefore refine future estimates while preserving the relation used for each earlier prediction and its assessment.

4 Experiments

4.1 Experimental Setup

We evaluate EffectPreconception in three iterative ML research settings. **Autoresearch** (Karpathy, 2026) is both a research harness and a benchmark for editing the training code of a small language model trained from scratch on one A100 GPU. Each search allows 100 nodes with 1,200 seconds of training per candidate. We report best retained validation bits per byte (BPB, lower is better). **MLE-bench Lite** (Chan et al., 2025) covers 22 Kaggle ML engineering competitions. We run the MLEvolve harness (Du et al., 2026) for 12 hours per competition and run each experiment independently three times, reporting submission, above-median, and medal rates. **PostTrainBench** (Rank et al., 2026) evaluates language-model post-training across seven benchmarks and four pretrained models. Each search uses the Autoresearch harness for 25 experiments with 1,200 seconds of training per candidate. We report the best test score on a 0–100 scale (higher is better). Detailed protocols are in Appendix A.

Baselines. We compare EffectPreconception with Autoresearch for language-model training and PostTrainBench, and with MLEvolve for MLE-bench Lite. We also include three prediction-based baselines on Autoresearch and MLE-bench Lite. (1) **ScorePred** directly predicts an edit's change in validation score and returns its directional prediction accuracy to subsequent proposals. (2) **LLAMBO** (Liu et al., 2024) is adapted to program edits, using LLM-based candidate generation and surrogate predictions to select an edit by expected improvement. (3) Our **FOREAGENT** (Zheng et al., 2026) adaptation uses candidate code and a data report to compare predicted performance between programs and select a candidate for execution. Appendix A describes these adaptations.

4.2 Autoresearch

Table 1 details final search performance after 100 nodes across both LLM backbones. EffectPreconception consistently achieves the lowest mean validation BPB, reaching 0.9404 BPB with Sonnet 5 and 0.9368 BPB with Opus 5. These results represent significant reductions of 0.0097 and 0.0115 BPB over standard Autoresearch, respectively. Furthermore, EffectPreconception outperforms the strongest prediction-based baseline on each LLM backbone, achieving a 0.0052 BPB reduction over FOREAGENT on Sonnet 5 and a 0.0082 BPB reduction over ScorePred on Opus 5. All evaluations report three-run averages under identical node and per-node training budgets. We report auxiliary LLM computational costs and a single-GPU H100 baseline comparison in Appendix D and Appendix C.2, respectively.

Table 2: **MLE-bench Lite results.** Percentages of the 22 competitions. We run each experiment independently three times with a 12-hour search budget per competition. AIDE and MLE-STAR results use a 24-hour budget (Nam et al., 2025) (Appendix A.5). Submission Rate counts valid submissions, and Any medal is the sum of the mutually exclusive gold, silver, and bronze rates. Bold marks the highest mean within each comparison block, including ties.

Method	Base LLM	Submission Rate ↑	Above median ↑	**Any medal** ↑	Medal breakdown		
					Gold	Silver	Bronze
AIDE	GPT-4o	65.7 ± 1.7	29.9 ± 1.6	18.6 ± 1.4	9.3 ± 1.0	5.8 ± 0.8	3.4 ± 0.6
AIDE	o1-preview	90.3 ± 1.6	58.2 ± 2.6	36.6 ± 2.6	20.7 ± 2.2	11.1 ± 1.7	4.8 ± 1.1
AIDE	Gemini 2.0 Flash	78.8 ± 5.0	39.4 ± 6.0	25.8 ± 5.4	12.1 ± 4.0	9.1 ± 3.5	4.5 ± 2.6
MLE-STAR	Gemini 2.0 Flash	95.5 ± 2.6	63.6 ± 6.0	43.9 ± 6.2	30.3 ± 5.7	4.5 ± 2.6	**9.1** ± 3.6
MLE-STAR	Gemini 2.5 Pro	**100.0** ± 0.0	**83.3** ± 4.6	**63.6** ± 6.0	**36.4** ± 6.0	**21.2** ± 5.1	6.1 ± 3.0
MLEvolve	DeepSeek V4 Flash	98.5 ± 2.6	74.2 ± 5.2	51.5 ± 2.6	40.9 ± 4.5	9.1 ± 4.5	1.5 ± 2.6
EffectPreconception		**100.0** ± 0.0	**83.3** ± 2.6	**63.6** ± 4.5	**43.9** ± 5.2	**12.1** ± 2.6	**7.6** ± 2.6
MLEvolve	Claude Opus 5	**100.0** ± 0.0	87.9 ± 5.2	77.3 ± 4.5	56.1 ± 2.6	10.6 ± 2.6	**10.6** ± 2.6
ScorePred		**100.0** ± 0.0	87.9 ± 6.9	77.3 ± 9.1	54.5 ± 4.5	**12.1** ± 6.9	**10.6** ± 2.6
LLAMBO		**100.0** ± 0.0	86.4 ± 4.5	71.2 ± 5.2	53.0 ± 2.6	9.1 ± 0.0	9.1 ± 4.5
FOREAGENT		**100.0** ± 0.0	87.9 ± 2.6	75.8 ± 2.6	57.6 ± 2.6	10.6 ± 5.2	7.6 ± 2.6
EffectPreconception		**100.0** ± 0.0	**89.4** ± 2.6	**78.8** ± 5.2	**65.2** ± 5.2	7.6 ± 5.2	6.1 ± 5.2

Beyond overall final performance, EffectPreconception demonstrates superior optimization dynamics across the entire search. To quantify total trajectory improvement, we define *recovery* as the initial-node validation BPB minus the best retained BPB at node 100 within a run. EffectPreconception achieves mean recovery scores of 0.0414 with Sonnet 5 and 0.0416 with Opus 5, exceeding all baseline methods on both LLM backbones. Crucially, this advantage manifests early in the search: with Sonnet 5, EffectPreconception reaches 0.9429 BPB at node 40, outperforming the final 100-node performance of every baseline before continuing its trajectory to node 100. Section 4.5 analyzes the contribution of explicit effect predictions to search performance.

4.3 MLE-BENCH LITE

As shown in Table 2, EffectPreconception substantially improves solution competitiveness across both LLM backbones under the same 12-hour search budget per competition. With DeepSeek V4 Flash, EffectPreconception increases the mean any-medal rate from 51.5% to 63.6% and the above-median rate from 74.2% to 83.3%. Across the 22 competitions, these gains correspond to approximately 2.7 additional medals and two additional above-median solutions per run.

When evaluated with Claude Opus 5, where default MLEvolve already achieves medals in over 77% of competitions, EffectPreconception's advantage concentrates at the competitive gold-medal threshold. EffectPreconception increases the mean gold-medal rate from 56.1% to 65.2%, while the overall any-medal rate rises modestly from 77.3% to 78.8%. This shift demonstrates EffectPreconception's ability to consistently produce higher-quality, better final delivered solutions. Across independent runs, EffectPreconception consistently earns 13 to 15 gold medals compared to 12 to 13 for MLEvolve, with its lowest gold count matching the highest count among all tested Opus 5 baselines. The few remaining competitions where Opus 5 fails to earn medals consist of highly specialized benchmarks, such as `new-york-city-taxi-fare-prediction` (requiring extensive, domain-specific tabular feature engineering) and `text-normalization-challenge-russian-language` (demanding complex linguistic representations), where performance bottlenecking stems from specialized data priors rather than iterative code modification.

Comparison against prediction-based baselines further confirms that predicting validation scores alone is insufficient for high-level search guidance. ScorePred, which provides feedback on predicted score direction accuracy, matches default MLEvolve's 77.3% any-medal rate but achieves a lower gold-medal rate (54.5%). Pre-execution candidate filtering approaches also yield suboptimal performance. LLAMBO and FOREAGENT achieve 71.2% and 75.8% any-medal rates, respectively, with FOREAGENT reaching a 57.6% gold-medal rate. Because all Opus 5 baselines achieve a 100% valid submission rate, these performance gaps directly reflect the quality of candidate programs. EffectPreconception achieves the top performance across both any-medal (78.8%)

Table 3: **PostTrainBench results.** Best retained scores (higher is better). Both methods use the Autoresearch harness. AR denotes the baseline and EP denotes EffectPreconception. Averages are over the four models; bold marks the higher score in each pair, both on a tie. The last row is the PostTrainBench leaderboard's weighted average.

	Qwen3-1.7B		Qwen3-4B		SmolLM3-3B		Gemma-3-4B		Average	
Benchmark	AR	EP	AR	EP	AR	EP	AR	EP	AR	EP
GSM8K	78.62	**79.00**	**90.07**	89.46	80.67	**81.58**	63.53	**64.44**	78.22	**78.62**
HumanEval	68.29	**69.51**	81.10	**81.71**	56.10	**56.71**	**42.68**	**42.68**	62.04	**62.65**
GPQA Main	33.93	**35.27**	38.17	**38.62**	31.47	**33.48**	30.13	**31.47**	33.43	**34.71**
AIME 2025	6.67	**10.00**	**16.67**	**16.67**	6.67	**10.00**	**3.33**	**3.33**	8.34	**10.00**
BFCL	**96.00**	**96.00**	97.00	**98.00**	**95.00**	**95.00**	91.00	**93.00**	94.75	**95.50**
Arena-Hard Writing	21.45	**24.04**	50.66	**53.27**	**40.86**	39.08	29.28	**31.00**	35.56	**36.85**
HealthBench	**24.58**	20.77	30.58	**32.40**	25.32	**26.53**	21.16	**23.91**	25.41	**25.90**
Weighted average	37.37	**38.12**	46.83	**47.59**	37.53	**38.94**	31.33	**32.53**	38.26	**39.29**

and gold-medal (65.2%) rates, demonstrating that combining pre-execution effect predictions with post-execution diagnostic checks provides superior search trajectory guidance over raw metric history or direct score forecasting.

4.4 POSTTRAINBENCH

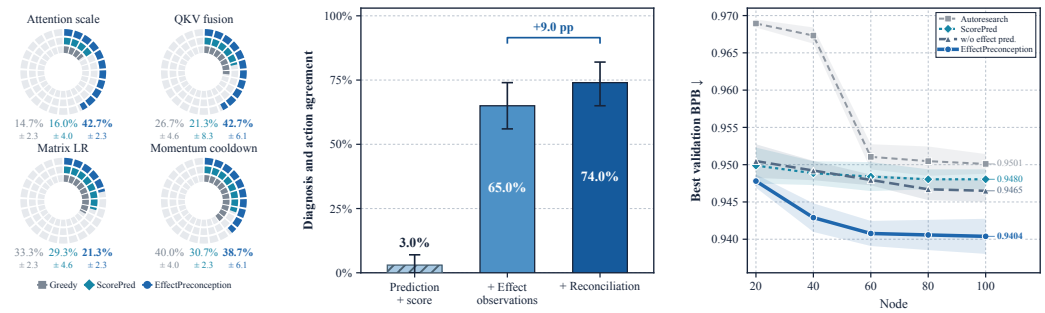
Table 3 summarizes PostTrainBench performance across seven benchmarks and four pretrained models. EffectPreconception improves the leaderboard's weighted score on every model, increasing the average from 38.26 to 39.29. The average score also improves on each benchmark, indicating that the overall gain extends across multiple tasks. Across the 20 comparisons on the five benchmarks evaluated without an LLM judge, EffectPreconception achieves 14 higher scores, five ties, and one lower score than Autoresearch. GPQA Main improves on all four models, raising the average from 33.43 to 34.71, while AIME 2025 increases from 8.34 to 10.00 through gains on Qwen3-1.7B and SmolLM3-3B. Improvements also extend to the two benchmarks evaluated by an LLM judge, Arena-Hard Writing and HealthBench, although neither improves on every model. Together, these results show consistent aggregate gains across models and benchmarks, with variation in the benefit for individual model and task combinations.

These results extend the evaluation of effect feedback from training models from scratch and developing ML pipelines to adapting pretrained models. In PostTrainBench, the agent searches over data mixtures, prompt formats, and optimization settings while the available training sets and benchmark evaluators remain fixed. Free-form predictions connect these edits to expected changes in measured model behavior, giving subsequent proposals evidence about the effects of earlier training choices alongside their benchmark scores. The improvements across models and benchmarks support using this feedback in post-training searches, where the agent refines how an existing model is trained for a target task. Appendix A.6 details this implementation, with additional prediction assessments and proposal traces in Appendices C.1 and E.7.

4.5 ABLATION STUDY

We hypothesize that checking an edit's predicted effect provides guidance for subsequent proposals beyond score feedback. To test this, we examine whether effect feedback improves proposal retention and how observations and reconciliation inform diagnosis. We then remove effect predictions while retaining observations to assess whether explicit predictions contribute beyond experimental history alone (Figure 4). All three ablations use Sonnet 5, with detailed protocols in Appendix B.

Effect Feedback and Proposal Retention. To evaluate whether effect feedback improves subsequent proposals, we compare retention across four starting cases spanning architecture (attention scaling and QKV fusion) and training dynamics (matrix learning-rate adjustment and momentum cooldown). Within each case, Greedy, ScorePred, and EffectPreconception begin with the same program and score feedback and evaluate 25 unrestricted proposals, retaining an edit only if it low-



(a) Retained edits (b) Effect-feedback diagnosis (c) Effect prediction

Figure 4: **Ablations of effect feedback and prediction.** (a) Retained-proposal rates across four starting cases. Rings show EffectPreconception, ScorePred, and Greedy from outside inward. (b) Diagnosis and next-action rule agreement on 100 Autoresearch edits. (c) Mean best retained validation BPB. Each experiment is run independently three times.

ers validation BPB relative to the current parent. EffectPreconception achieves the highest retention in both architectural starting cases, retaining 42.7% of proposals in each (Figure 4(a)). Its advantage is less consistent in the training-dynamics cases, where it performs similarly to Greedy and better than ScorePred after momentum cooldown, but falls below both baselines after matrix learning-rate adjustment. These mixed results suggest that effect prediction for training dynamics may be more challenging, motivating future work on how agents formulate and test predictions in these settings.

Diagnostic Impact of Observations and Reconciliation. To evaluate how observations and reconciliation guide diagnosis, we conduct a paired offline ablation on recorded Autoresearch edits. Holding edit descriptions, predictions, and scores fixed, we add observations and then reconciliation. Responses must match the predefined diagnosis and action rule and pass a grounding check. Without observations, 100% of responses request measurements as prompted, yielding 3.0% agreement. Adding observations raises agreement to 65.0%, and adding reconciliation over identical observations further increases it to 74.0% (Figure 4(b)). These gains confirm that post-execution observations and reconciliation are essential for accurate agent diagnosis and actionable failure recovery.

Contribution of Effect Prediction to Search Performance. To isolate the contribution of effect prediction during search, we remove registered effect cards and free-form predictions while retaining measured history and outcome relations. Both variants can therefore use earlier measurements, while the full framework additionally records whether each edit produced its predicted effect. The full framework achieves a mean final validation BPB of 0.9404, compared with 0.9465 for the variant without effect prediction (Figure 4(c)). The variant without effect prediction still outperforms ScorePred and Autoresearch, indicating that the retained feedback provides useful guidance on its own. The additional improvement from explicit predictions supports pairing measured history with expectations recorded before execution, so subsequent proposals can also use evidence about whether earlier edits behaved as predicted.

5 Conclusion

EffectPreconception improves iterative ML research by pairing validation performance with evidence about whether an edit produced its intended effect. Across Autoresearch, MLE-bench Lite, and PostTrainBench, EffectPreconception improves the quality of the models and solutions found by existing agents. In Autoresearch, removing effect predictions while retaining measured history and outcome relations increases mean final validation BPB from 0.9404 to 0.9465, supporting the value of comparing observations with expectations stated before execution.

The accumulated evidence could also guide which experiments an agent chooses to run next. EffectPreconception preserves what an experiment reveals about an agent's expectations and carries that evidence into subsequent proposals. While our current implementation assesses predictions about individual edits, future work could use accumulated evidence to design follow-up experiments that distinguish competing explanations for an observed result. This points toward autonomous research in which each experiment informs both the next model and the next hypothesis.

AI USE STATEMENT

We used generative AI tools to retrieve related literature, assist with baseline code adaptations, generate the case study figures, and revise the method figure layout. Additionally, we used generative AI tools to assist with grammar checks and polishing of the draft. We have reviewed all AI-assisted work. The authors carefully reviewed the code throughout the adaptation process, checked the cited sources, and checked the polished text and figures against the experimental records. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work evaluates automated ML research on existing benchmarks. Greater research automation may broaden access to model development and reduce repetitive human effort, while also lowering the barrier to developing systems for harmful applications. Models developed with these agents may inherit biases or privacy risks from their training data, with consequences for people affected by their use. Downstream applications therefore require consideration of fairness, privacy, and potential harm, particularly in settings where model outputs inform decisions about people.

REPRODUCIBILITY STATEMENT

Sections 3 and 4.1 describe the method and experimental setup. Appendices A and B provide task-specific implementations, baseline adaptations, and ablation protocols. The anonymous code repository is linked in the abstract.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Yash Akhauri, Xingyou Song, Arissa Wongpanich, Bryan Lewandowski, and Mohamed S. Abdelfattah. Regression language models for code. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.26476.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950. URL https://journals.ametsoc.org/view/journals/mwre/78/1/1520-0493_1950_078_0001_vofeit_2_0_co_2.xml.

Paul Brunzema, Louis Tiao, Nhat Le, Kevin De Angeli, Yao Xuan, and Djordje Gligorijevic. Agentic Bayesian optimization through surrogate-augmented autoresearch. *arXiv preprint arXiv:2608.00316*, 2026. URL https://arxiv.org/abs/2608.00316.

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

Yuhan Chen, Zhihua Tian, Mahavir Dabas, Charith Peris, Rahul Gupta, Ming Jin, Feiyang Kang, Siyuan Zhang, Nan Wang, and Ruoxi Jia. Belief-calibrated optimization: An explicit world model for agentic optimization. *arXiv preprint arXiv:2609.01861*, 2026. URL https://arxiv.org/abs/2609.01861.

A. P. Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2): 278–290, 1984. doi: 10.2307/2981683. URL https://academic.oup.com/jrsssa/article/147/2/278/7106293.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-CLIMB: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *Advances in Neural Information Processing Systems*, volume 38, pp. 29096–29124, 2025. doi: 10.52202/085713-0865. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/24a8968affe71ffe4067d022b9d16566-Paper-Datasets_and_Benchmarks_Track.pdf.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, Zichen Liang, Boyuan Sun, Tianshuo Peng, Yifan Zhou, Xin Li, Jie Zhou, Liang He, Bo Zhang, and Lei Bai. MLEvolve: A Self-Evolving Framework for Automated Machine Learning Algorithm Discovery. *arXiv preprint arXiv:2606.06473*, 2026. URL https://arxiv.org/abs/2606.06473.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030, 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating Language Agents on Machine Learning Experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

Ganesh Jawahar, Muhammad Abdul-Mageed, Laks V. S. Lakshmanan, and Dujian Ding. LLM performance predictors are good initializers for architecture search. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10540–10560. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.627. URL https://aclanthology.org/2024.findings-acl.627/.

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. URL https://arxiv.org/abs/2502.13138.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147. URL https://doi.org/10.1023/A:1008306431147.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Andrej Karpathy. Autoresearch. GitHub repository, 2026. URL https://github.com/karpathy/autoresearch. Accessed September 5, 2026.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large Language Models to Enhance Bayesian Optimization. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=OOxotBmGol.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan Ö. Arık, and Tomas Pfister. MLE-STAR: Machine Learning Engineering Agent via Search and Targeted Refinement. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a9619dd0f0d54a5cf7734add1dc38cd1-Abstract-Conference.html.

Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018. doi: 10.1073/pnas.1708274114. URL https://doi.org/10.1073/pnas.1708274114.

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. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25140.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *Advances in Neural Information Processing Systems*, volume 38, pp. 100092–100118, 2025. doi: 10.52202/085713-3345. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/904e89bb4e632e75fb47f093b620b257-Paper-Conference.pdf.

Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can LLM agents automate LLM post-training?, 2026. URL https://arxiv.org/abs/2603.08640.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, volume 37, pp. 68658–68685, 2024. doi: 10.52202/079017-2193. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7ede97c3e082c6df10a8d6103a2eebd2-Paper-Conference.pdf.

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*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Howe Tissue, Venus Wang, and Lu Wang. Scaling law with learning rate annealing. *Advances in Neural Information Processing Systems*, 38:91017–91053, 2026.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Kelvin Niu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H. Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Stenetorp, Nicola Cancedda, Jakob Nicolaus Foerster, and Yoram Bachrach. AI Research Agents for Machine Learning: Search, Exploration, and Generalization in MLE-bench. *arXiv preprint arXiv:2507.02554*, 2025. URL https://arxiv.org/abs/2507.02554.

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

Zherui Yang, Fan Liu, and Hao Liu. DSWorld: A Data Science World Model for Efficient Autonomous Agents. *arXiv preprint arXiv:2607.15901*, 2026. URL https://arxiv.org/abs/2607.15901.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Jingsheng Zheng, Jintian Zhang, Yujie Luo, Yuren Mao, Yunjun Gao, Lun Du, Huajun Chen, and Ningyu Zhang. Can We Predict Before Executing Machine Learning Agents? In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3941–3974. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.182. URL https://aclanthology.org/2026.acl-long.182/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

APPENDIX CONTENTS

A Method and Implementation Details **15**

A.1 Terminology . . . 15

A.2 Research cycle . . . 15

A.3 Interpreting reconciliation . . . 16

A.4 Autoresearch . . . 17

A.5 MLE-bench Lite . . . 20

A.6 PostTrainBench . . . 21

A.7 Baselines . . . 23

B Additional Experimental Setup **23**

B.1 Effect feedback and proposal retention . . . 24

B.2 Effect observations and reconciliation . . . 24

B.3 Effect prediction during search . . . 25

C Additional Results and Analysis **25**

C.1 Effect prediction assessments on PostTrainBench . . . 25

C.2 Comparison with the published Autoresearch H100 run . . . 25

D Computational Cost and Overhead **26**

E Case Studies **27**

E.1 Using effect feedback to refine attention gating . . . 27

E.2 Learned attention scaling . . . 27

E.3 Optimization and attention edits . . . 28

E.4 Adaptive attention and training changes . . . 28

E.5 Different hypotheses from a shared parent . . . 28

E.6 Effect evidence beyond the validation score . . . 29

E.7 PostTrainBench episodes . . . 29

E.8 Worked example of an evidence ledger entry . . . 29

F Limitations **39**

A METHOD AND IMPLEMENTATION DETAILS

This appendix expands the terminology, research cycle, and reconciliation procedure in Section 3, then describes their implementation in each task and the baseline adaptations. Across tasks, predictions are recorded before execution and effects are assessed without the final validation score, while the prediction forms and measurements reflect the task. Autoresearch combines registered effect cards with free-form predictions, MLE-bench Lite attaches one effect prediction and one measurement to each pipeline edit, and PostTrainBench uses free-form predictions without outcome relations.

A.1 TERMINOLOGY

EffectPreconception expresses an agent's preconception, an expectation about an edit's effect formed before execution, as a testable prediction. By specifying an observable, a comparison reference, and an expected change or pattern, this prediction gives the framework a concrete expectation to assess against subsequent measurements. The assessment is recorded alongside the validation score, with a supported outcome relation providing an additional estimate of the performance change associated with the effect when available. The ledger preserves these expectations, measurements, and assessments together with the performance outcome, so each experiment can inform subsequent proposals. Table 4 summarizes the terms used for these records.

Table 4: Terminology. Terms used throughout the appendix.

Term	Meaning
Preconception	An expectation about an edit's effect formed before execution.
Effect	A measurable change an edit produces in the model or training process, assessed through a specified observable and comparison reference.
Effect prediction	A testable statement of the change or pattern expected from an edit, fixed before execution.
Registered effect card	An effect prediction on a quantity from a list the task defines, such as the loss on long documents, with a predicted direction (up, down, or no change) and an optional size.
Free-form prediction	An effect prediction whose quantity is named by a separate model call, together with a short probe that measures it.
Held / Refuted	Valid measurements support / do not support the specified prediction under its assessment rule.
Not observable	The measurement is missing or invalid, so the prediction cannot be judged.
Outcome relation	An optional mapping from a predicted or measured effect to an expected change in validation score.
Evidence ledger	The record of every prediction with its measurement and verdict; the agent reads a summary of it before each new proposal.

A.2 RESEARCH CYCLE

Each iteration in Algorithm 1 connects a new proposal to the evidence accumulated so far. After the base policy selects a parent x_p, the agent builds its context h_t from that parent, earlier experiments, and the ledger $\mathcal{L}_{t-1}$, then proposes a candidate x_t with prediction record C_t. The framework completes this record with any supported outcome relations and performance estimates, fixing both the predictions and their measurement definitions before execution. Each assessment can therefore refer back to the original expectation, and the current outcome cannot enter the relation used to predict it.

Once execution returns measurements r_t and a score y_t, the effect assessment a_t checks the prediction against r_t and, where required, parent measurements r_p, independently of the proposer's rationale and validation score. Any performance predictions are assessed separately against

Algorithm 1 EffectPreconception research cycle

Inputs Initial evaluation, budget, base policy, measurement protocol

Initialize evidence ledger $\mathcal{L}_0$ from the initial evaluation
for $t = 1, 2, \ldots$ within budget **do**
    Select parent x_p with prior evaluation (r_p, y_p)
    Build h_t from parent, history, and summary of $\mathcal{L}_{t-1}$
    $(x_t, C_t) \leftarrow \textsc{Propose}(h_t)$
    Complete C_t with supported outcome relations and estimates
    Fix C_t and its measurement definitions
    $(r_t, y_t) \leftarrow \textsc{Evaluate}(x_t)$
    $a_t \leftarrow \textsc{AssessEffects}(C_t, r_t, r_p)$
    Assess performance predictions in C_t against $\Delta y_t = y_t - y_p$
    Record $(x_p, x_t, C_t, r_t, y_t, a_t)$ and outcome assessments in $\mathcal{L}_t$
    Update future relations from eligible experiment pairs
    Retain or discard x_t under the base policy
end for
return implementation selected by the base policy

$\Delta y_t = y_t - y_p$, and the ledger records these assessments with the observed outcome even when the candidate is discarded. This evidence becomes available to the next proposal and, after assessment, can refine future outcome relations while preserving the relation associated with each earlier prediction. Candidate retention and final selection continue to follow the base policy.

A.3 Interpreting reconciliation

Reconciliation helps the agent decide which expectation to revisit by considering the effect assessment alongside the performance outcome. A held prediction indicates that measurements support the specified change or pattern, although it does not establish that the effect caused the score change. A refuted prediction may arise from an opposite change, movement below measurement resolution, or partial agreement with a predicted pattern, so it calls for examining which expectation failed. These distinctions motivate the follow-up investigations in Table 5, which covers valid measurements and score changes whose magnitudes exceed the task's no-change threshold.

Table 5: Possible follow-ups from the effect verdict and performance change.

Effect verdict	Performance improves	Performance worsens
Held	Measurements support the predicted effect and performance improves. Test whether the benefit persists.	Measurements support the predicted effect but performance worsens. Investigate whether other changes or costs offset its expected benefit.
Refuted	Performance improves, but measurements do not support the effect prediction. Reconsider the prediction and examine other changes.	Performance worsens and the effect prediction is unsupported. Investigate which expectation failed and how the edit could be revised.

When performance does not change discernibly, the effect assessment can still distinguish a supported expectation from one that needs reconsideration, providing evidence for the next proposal even when the score offers little guidance. Missing or invalid measurements require a different response, since the prediction is not observable and usable evidence is needed before it can be interpreted. These considerations guide subsequent proposals while leaving candidate retention to the base policy. The task implementations below specify the measurement rules, thresholds, and feedback used to apply them.

A.4 AUTORESEARCH

A.4.1 SETUP

In Autoresearch, the agent iteratively edits a language-model training program and evaluates candidates using validation bits per byte (BPB), with lower values indicating better performance. We therefore set $y = -\mathrm{BPB}$ to follow the convention in Section 3 that higher scores are better. The base LLM specified in Table 1 proposes each edit and writes its registered effect card, while separate calls to the same model generate and assess the free-form predictions.

The starting program is implemented in PyTorch (Paszke et al., 2019) and combines rotary position embeddings (Su et al., 2024), RMS normalization (Zhang & Sennrich, 2019), and FlashAttention-3 (Shah et al., 2024). It is trained on Autoresearch's shuffled ClimbMix shards (Diao et al., 2025), using Muon-style updates for the transformer weight matrices (Jordan et al., 2024) and AdamW for the embeddings, output head, and scalar parameters (Loshchilov & Hutter, 2019). During search, the agent may modify the architecture, optimizer, and training settings, while data preparation and evaluation remain fixed.

Each search has a budget of 100 nodes, including the initial program and any failed or early-terminated runs. Within this budget, each candidate is trained from scratch on one GPU for at most 1,200 seconds and is retained if it lowers validation BPB. The time limit covers training alone, with agent calls and post-training measurements taking additional time. Appendix D reports the auxiliary LLM tokens used by prediction and assessment calls.

A.4.2 EFFECT PREDICTIONS AND MEASUREMENTS

Before execution, an edit can be described by one or more registered effect cards, each specifying an observable and the predicted direction and optional magnitude of its change from the parent. Registered cards provide the effect predictions used by supported outcome relations to estimate score changes. A separate free-form predictor reads the proposed edit, its rationale, the code diff, and the parent's diagnostics to define another observable, write a measurement-only probe, and specify the expected pattern in its output. When existing diagnostics cannot capture that pattern, the predictor can also add a monitor to record a signal during training. The expected pattern, probe, and optional logical test are all saved before execution, and the free-form prediction receives its own assessment. The logical test remains a diagnostic and does not determine that verdict.

Alongside these effect predictions, the proposer supplies an auxiliary prediction of resource use and the probability and direction of a discernible score change. An informative resource prediction can also supply a registered card when the proposer has not written one, whereas a card for a replication or another experiment set up in advance describes its planned comparison. To distinguish these sources, each card records who authored it, and only cards written by the proposer count toward its prediction accuracy.

To support a numeric outcome estimate, the framework can refine or supply a prediction's magnitude using the optional procedures in Table 6. Refinement asks the agent to specify or reconsider the magnitude before execution, whereas imputation supplies a missing value and stores it separately from the agent's prediction. The agent's predicted direction is assessed in either case, while an imputed magnitude is never attributed to the agent. Resource and schedule estimates retain their recorded magnitudes or their defaults.

To assess the recorded predictions, we measure changes using the rules and movement thresholds for Autoresearch's 20 registered observables in Table 7. To make parent and child measurements comparable, all loss components use the same fixed sample of training data with identical inputs and grouping rules. For position-based groups, a token's offset is measured from the start of its document, or from the row's start if the document began in an earlier row; document-length groups likewise use only the portion visible within the row. Each grouping partitions the full sample, with one additional observable measuring its overall mean loss. Changes whose magnitudes fall below the relevant movement threshold are treated as no change, while larger changes are classified as increases or decreases.

Registered probes evaluate training examples outside the timed training without changing model weights, and they restore the random-number state afterward. Free-form probes likewise leave the

Table 6: Optional rules for the size of an Autoresearch prediction.

Procedure	Rule
Magnitude refinement	If a loss-component or logit-margin prediction expects a change but gives no size, ask once for the size. If the predicted size is above the 95th percentile of earlier measured changes of that quantity, ask once to revise or confirm it; this check needs at least five earlier judged predictions with valid measurements.
Magnitude imputation	For a loss component, use the median absolute change measured after earlier judged agent predictions, when there are at least three. Otherwise use the measurement resolution. For a logit margin with no size, use a relative change of 2%.
Reconsideration request	Ask the proposer to reconsider the edit when its auxiliary prediction says a discernible score change is unlikely and it gives no informative registered card.

Table 7: Autoresearch's 20 registered observables and their movement thresholds. Each bin includes its lower bound and excludes its upper bound. Frequency rank starts at zero. z_p is the parent's measurement.

Observable group	Measurement	Movement threshold
Model size (1)	Parameter count	0.2% relative
Optimization work (1)	Completed optimization steps	3.7% relative
Execution rate (1)	Tokens processed within the training budget; throughput so far if the run stops early	3.7% relative
Training curve (1)	Mean loss over the final five 10-step bins	0.004 nats
Position losses (6)	Token positions with boundaries $0, 64, 128, 256, 512, 1024, \infty$	$\max(0.005, 0.002\lvert z_p\rvert)$ nats
Frequency losses (4)	Target-token frequency ranks with boundaries $0, 64, 512, 2048, \infty$	Same loss threshold
Document-length losses (4)	Visible lengths with boundaries $0, 256, 768, 1280, \infty$ tokens	Same loss threshold
Overall sample loss (1)	Mean loss over all tokens in the sample	Same loss threshold
Logit margin (1)	Mean difference between the largest and second-largest logits	$0.005\lvert z_p\rvert$

weights unchanged and may use held-out data when the task permits. Because validity is checked for each measurement independently, a deliberately stopped run can still provide a usable execution rate even when no final score is available, while summaries and fits omit any missing values.

A.4.3 Outcome relations

An outcome relation maps a predicted effect to an expected change in validation BPB using the construction appropriate to its observable (Table 8). Relations fitted from experiment history use only earlier valid results, including kept and discarded candidates. Each relation is fixed before the current edit runs so that its assessment uses the original estimate, after which the result can inform later relations.

For the logit margin, e is the child's margin minus the parent's, divided by the absolute parent margin. We fit a least-squares line with an intercept to earlier effect–score-change pairs $(e_i, \Delta y_i)$. The outcome relation uses the fitted slope b with its intercept set to zero:

$$M_t(e) = b\, e. \tag{4}$$

This construction predicts no score change when the effect is zero. A pair is used only when the parent and child measurements and the final score are all valid.

For a loss component, the sample's mean loss is the average of its groups, weighted by the number of tokens in each. If the chosen group holds a share w of the tokens and its mean loss changes by e

Table 8: How each Autoresearch outcome relation is built, and the data it needs.

Observable	How it is built	Data it needs
Logit margin	A linear slope fitted to relative margin changes and score changes	At least 10 valid pairs, effects that are not all equal, and a positive slope
Sample loss component	The share of the sample loss that the group carries, converted to BPB	Valid parent loss, validation measurements, and token share
Size or work	An inverse power-law fit to model size and training work	At least 8 results and 4 different size–work pairs
Training-curve endpoint	A fit to the parent's training curve, converted to BPB	A valid curve fit, and either 3 measured schedule changes or a positive conversion from 3 of the parent's training–validation pairs

nats, and the other groups do not change, then

$$M_t(e) = -w \frac{\mathrm{BPB}_p}{\ell_{\mathrm{val},p}} e, \tag{5}$$

where the parent's validation BPB and validation loss $\ell_{\mathrm{val},p}$ convert nats to BPB. The minus sign comes from $y = -\mathrm{BPB}$. For the overall sample loss, $w = 1$. A relative loss-change prediction is first converted to nats by multiplying it by the parent's loss for that group. The token-weighted decomposition of loss is exact on the measurement sample; using it to predict validation BPB assumes that the group's loss change transfers to validation data and that changes in other groups do not offset it.

For resource effects, we fit the scaling law of Hoffmann et al. (2022) to validation BPB: $L(N, D) = c + aN^{-\alpha} + bD^{-\beta}$, where N is the parameter count and D is the number of processed tokens. A second form, for schedule changes, replaces D with training work weighted by the learning rate and adds a linear annealing term. We select the exponents α and β by grid search, fitting the remaining coefficients to earlier results by least squares and retaining the fit with the lowest squared error. Each exponent is drawn from $\{0.05, 0.1, 0.2, 0.35, 0.5, 0.75, 1, 1.5\}$. The expected improvement is the predicted parent BPB minus the predicted child BPB. A throughput or step prediction changes the training-work input while holding the parameter count fixed; a parameter-count prediction changes the parameter-count input while holding training work fixed. An edit that changes both inputs requires a prediction for each.

For schedule edits, we adapt the learning-rate annealing law of Tissue et al. (2026) by fitting the parent's training curve as $\ell = c + aS_1^{-\alpha} - bS_2$, where S_1 is the area under the learning-rate curve so far and S_2 adds up learning-rate decreases with momentum 0.99 over 10-step bins. The child's schedule then gives a predicted change in the final training loss. To convert the predicted training-loss change to a BPB change, we fit a linear conversion with zero intercept to earlier schedule changes and clip its slope to $[0, 1]$. With fewer than three earlier schedule changes, the conversion can instead use a positive slope fitted to at least three paired training and validation measurements from the parent. Without a valid conversion there is no numeric estimate.

A.4.4 Assessment and feedback

A registered card is judged by its direction: we compare the predicted direction with the measured one, using the thresholds in Table 7. When both directions are the same and not zero, and the card gave a size, we also record the measured size divided by the predicted size. A free-form prediction is judged by a separate evaluator, which sees only the observable, the expected pattern, and the measurements. The evaluator returns held when the measurements clearly support the expectation, refuted when they contradict it or do not support it, and not observable when there are no usable measurements. A vague expectation is refuted. So refuted means only that the measurements do not support the prediction; the opposite effect need not have occurred.

The auxiliary score forecast is assessed separately from the effect prediction. We use a Brier score (Brier, 1950) for its predicted probability of a discernible score change and assess its direction only when the magnitude of the measured change reaches 0.0006 BPB, twice the measured

noise of 0.0003 BPB. Resource predictions are assessed against measured resource use and whether the run finishes. When an outcome relation supports both the predicted and measured effect, Equation 3 further separates the score-estimation error into contributions from the effect prediction and the relation.

These assessments enter the evidence ledger alongside each prediction's author, parent, measurements, score change, and outcome estimate. The agent receives separate summaries for registered cards and free-form predictions. For registered cards, the summary counts held and judged cards for each observable. For free-form predictions, the summary reports the numbers of held and assessed predictions across all edits, adding counts by edit type once a type has at least two assessed predictions. A longer summary can add, for each observable with at least three results, the median measured change and the ratio of the median predicted size to the median measured size, and the three most recent judged cards with their text, verdict, and measured change. The proposer reads these summaries together with its task, the parent, and the history of experiments. The free-form predictor reads the proposed edit and the parent's diagnostics, and the evaluator reads only the prediction and the measurements.

A.5 MLE-bench Lite

A.5.1 Setup

The 22 competitions cover image, text, audio, and tabular data, requiring the agent to prepare data, train and tune models, and produce a submission file. We run MLEvolve with DeepSeek V4 Flash and Opus 5, allocating 12 hours of search per competition. During search, the agent evaluates candidates on its own validation split, while the benchmark grades the final submission on its held-out test set. We aggregate these grades over all 22 competitions to compute submission, above-median, and medal rates. The any-medal rate counts valid submissions reaching at least bronze, with gold, silver, and bronze reported as mutually exclusive categories. Each experiment is run independently three times.

Reference rows. Table 2 also lists AIDE and MLE-STAR as published by Nam et al. (2025), who use the benchmark's 24-hour limit per competition. The results use three seeds for MLE-STAR and for AIDE with Gemini 2.0 Flash. AIDE with o1-preview and GPT-4o uses 16 and 36 seeds, respectively; Nam et al. (2025) take these rows from Chan et al. (2025).

A.5.2 Effect predictions and measurements

In MLEvolve's improvement, evolution, and fusion stages, the agent records an effect prediction after code review and before the edit runs. The predictor reads the edit, the reason for it, the code diff, and the parent's diagnostics. It then picks one observable that shows whether the intended effect occurred. The observable cannot be the validation metric or a direct function of it. The prediction gives a direction, the reference it is compared with, and a size where one applies.

To assess this prediction, the measurement probe uses the candidate's predictions on the validation split or its logged diagnostics over time. A probe can measure, for example, how often two models disagree, how each class behaves, or classification calibration, which compares the model's confidence with how often it is right (Guo et al., 2017). The probe measures parent and child in the same way when both are available. A numeric change counts only when its magnitude exceeds the movement threshold $\max(10^{-4}, 0.002|z_p|)$, where z_p is the parent's measurement. Agreement between the predicted and measured magnitudes is recorded separately. When a numeric check is not possible, a separate evaluator judges the prediction as held, refuted, or not observable, using the rules in Appendix A.4.4.

A.5.3 Outcome relations

The measured effects also supply the history used to estimate score changes. For each competition and observable, we fit an outcome relation between the measured effect and the change in the agent's validation score. The fit uses up to 20 recent finished experiments, both kept and discarded, whose parent and child measurements and scores are valid. Two experiments can share a fit only when they use the same measurement rule, unit, validation split, subset of classes or samples, and validation

Table 9: Conditions for a numeric MLEvolve outcome estimate. They are checked before the edit runs. The error split after the run also needs the measured effect to lie in the same range.

Requirement	Check
History	At least 5 compatible pairs, using at most the 20 most recent.
Effect variation	The standard deviation of the measured effects is larger than the median movement threshold of the pairs.
Support	The predicted effect lies between the smallest and the largest measured effect.
Usable prediction	A change from the parent with a size, plus the parent measurement when the units must be converted. A no-change prediction needs no size; its effect is zero.

metric. The same observable name or kind of edit is not enough. Each experiment counts once. Final benchmark scores and medals are never used.

The relation takes the effect e, the child's measurement minus the parent's, and predicts the change in validation score, with higher meaning better. We fit a line and shrink its slope toward zero, keeping predictions closer to the historical mean score change when few experiments are available. For n usable pairs, let b be the least-squares slope of a line with an intercept, and let $\bar{e}$ and $\overline{\Delta y}$ be the mean effect and the mean score change. The relation is

$$M_t(e) = \overline{\Delta y} + \frac{n}{n+\lambda}\, b\,(e - \bar{e}), \tag{6}$$

with $\lambda = 5$ by default. Shrinkage preserves the predicted score change $\overline{\Delta y}$ at the mean effect $\bar{e}$, while reducing sensitivity to departures from that mean. The slope can be positive or negative, and the intercept can be nonzero because the edit may affect performance through changes not captured by the chosen observable.

Before the edit runs, the predicted direction and magnitude determine the effect estimate $\hat{e}_t$. A relative magnitude is converted to the observable's units by multiplying it by the absolute parent measurement. We use the relation only when the conditions in Table 9 hold. Before execution, we record the fitting history, measurement rule, coefficients, supported effect range, and expected score change, which remain fixed for the current assessment. When there is no estimate, the effect prediction is still judged.

A.5.4 Assessment and feedback

After the edit runs, reconciliation assesses the effect prediction without the validation score and separately compares the expected and measured score change. When the measured effect lies within the outcome relation's supported range, Equation 3 separates the error into contributions from the effect prediction and the relation. Otherwise, we retain the original estimate and effect verdict without decomposing the error. The completed experiment enters the fitting history only after this assessment, preserving the distinction between evaluating a relation and updating it. Direct score predictions are assessed independently and do not replace a missing outcome relation.

The evidence ledger links each prediction and assessment to the candidate and its parent. Before the next proposal, the agent receives the chosen parent's verdicts and the three most recent assessed experiments in the competition, with each entry containing the predicted and measured effect, the verdict, the expected and measured score change, and any error decomposition. The predictor also receives its past prediction accuracy for each observable. For each outcome relation, the summary reports the number of fitting pairs and the supported effect range, together with the number and mean absolute error of assessed estimates among the last 20 sharing the measurement rule. Following the sequential evaluation of Dawid (1984), this error includes only estimates recorded before their outcomes were known.

A.6 PostTrainBench

A.6.1 Setup

PostTrainBench comprises seven benchmarks (GSM8K, HumanEval, GPQA Main, AIME 2025, BFCL, Arena-Hard Writing, and HealthBench) and specifies four base models (Qwen3-1.7B-Base,

Qwen3-4B-Base, SmolLM3-3B-Base, and Gemma-3-4B-PT). We evaluate both methods on all 28 benchmark–model pairs in this suite. The first five benchmarks use exact match or unit tests, while Arena-Hard Writing and HealthBench use a model judge. We use `gpt-5.6-luna` as the judge for both methods. The fixed evaluator scores each candidate on the full test set, producing a score y from 0 to 100, with higher values indicating better performance. For each model, we combine its seven benchmark scores using the leaderboard weights, which sum to one, and report the mean weighted score across the four models.

Each search runs 25 experiments, including the initial program, with every candidate trained from the base weights for 1,200 seconds. Experiments that crash or make no edit count toward this budget and receive no score. Sonnet 5 proposes edits, writes free-form effect predictions, assesses those predictions, and produces lessons through separate calls with inputs appropriate to each role. This implementation uses free-form predictions without registered effect cards or outcome relations, so reconciliation compares predicted effects with measured effects, while performance is evaluated by the benchmark's fixed evaluator.

The agent develops candidates by editing the settings in `train.py`, including the mixture weights of fixed training sets, the proportions of three prompt formats, the number of training samples, sequence length, and optimizer, learning-rate schedule, and batch settings. The script constructs the data mixture, fine-tunes the base model, saves the result, and invokes the evaluator. The agent can read all files but edit only `train.py`, with the available training sets, evaluator, chat template, and decoding settings held fixed throughout the search.

A.6.2 Effect predictions and measurements

After each edit and before training, the effect predictor writes one prediction. It reads the change, the problem and the mechanism the agent states, the diff, the parent's diagnostics, and the values of the parent's own probe. It returns an observable, a probe that measures it, the expected effect in words, and a true-or-false test on the probe's output. The observable must reflect the stated mechanism and must be measured on the held-out split, or compare training and held-out data. Training dynamics, such as the training-loss curve or gradient norms, can support the reasoning but cannot be the observable. The observable can use this run's measurements alone or compare them with the parent's. The predictor also sees its five most recent judged predictions, each with the change, observable, expected effect, measured values, verdict, and the evaluator's reason. A separate score predictor reads the parent's score, the change, the reason for it, and the diff. It returns a direction (up, same, or down) and a rough size in points.

Every run logs the training loss, gradient norm, learning rate, and fraction of time used at each logging step. After training, it computes eight numbers over the completion tokens: the held-out negative log-likelihood on 96 held-out rows, the loss on 96 training rows, the gap between the two, the prediction entropy, the logit margin, the logit norm, the token accuracy, and a coarse calibration error. The held-out rows come from 512 examples set aside from the task's own training data and never used for training. Only the evaluator reads the test set. The probe written by the effect predictor runs once after training, on the final model and one held-out batch, and returns numbers. An optional monitor can log one more signal at each step during training, but it cannot run the model. Neither one changes the model, and both run outside the timed training.

A.6.3 Assessment and feedback

After training, an evaluator sees only the observable, the expected effect, the probe's values, and the values of the parent's probe. It returns held, refuted, or not observable, with a reason. An empty probe output is not observable. The true-or-false test is computed as a diagnostic and does not decide the verdict. The score prediction is judged only when the score changes by at least 1.0 point, the smallest change the evaluator repeats reliably. It is held when its direction matches the change and refuted otherwise; a smaller change makes it not observable. A lesson call then reads the change, the reason for it, the prediction, the verdict, and the parent and child scores. It returns a failure category, a mechanism type, and a one-line lesson.

After each judged experiment, a follow-up call writes a short recommendation for the next edit. It reads the stated problem, the change, the mechanism, the observable, the expected effect, the measured values, the verdict with its reason, and the parent and child scores. The recommendation

does not decide what is kept: the search keeps a child only when its score is higher than the best score so far.

To form the next proposal, the agent receives this recommendation together with the current best score, the parent's diagnostics, the twelve most recent experiments, and the evidence ledger grouped by mechanism family. There are six families: corpus mixture, prompt format, loss masking, data shape, learning-rate schedule, and batch; each mechanism type the lesson call writes is mapped to one of them. For each family, the ledger reports the number of held predictions and score-improving edits among its last 8 assessed predictions. It also reports these counts over the last 16 predictions across all families. With fewer than 2 records, it says the evidence is insufficient. The proposer also sees a table of the settings it has changed, each with how often it was tried and its last three score changes.

A.7 Baselines

We implement the baselines within the task harnesses used by EffectPreconception. On Autoresearch, the base LLM listed in Table 1 generates edits and predictions through Claude Code. Each selected candidate modifies the best `train.py` found so far, trains from scratch within the training budget, and replaces the parent when its validation BPB is lower. On MLE-bench Lite, Opus 5 generates edits and predictions through the Claude Agent SDK. LLAMBO and FOREAGENT generate multiple candidate children after MLEvolve selects a parent, applying their selection procedure within the improvement stage while preserving the other stages. Only the selected child is executed and assigned a measured score; unselected children are removed from the search tree.

ScorePred. ScorePred replaces effect prediction with a direct forecast of the change in validation score, which the proposer supplies with each edit before execution. After evaluation, the forecast is assessed against the measured score change, and the resulting assessments provide feedback on the accuracy of earlier forecasts for subsequent proposals. This feedback concerns predicted performance rather than an observable effect of the edit. ScorePred uses the same task harness and research budgets as EffectPreconception.

LLAMBO. We adapt LLAMBO's discriminative variant (Liu et al., 2024) to candidate code edits, representing each candidate by its plan and code diff relative to the parent. In both tasks, the method generates four candidates and conditions its surrogate predictions on the task, parent code, and up to 20 previous designs with measured scores. On Autoresearch, the target is the best BPB so far minus 10% of the observed score range. On MLE-bench Lite, the history is drawn from the same competition, and the target follows the competition metric's optimization direction. Once three valid scores are available, the surrogate produces eight score predictions per candidate across two random orderings of the history, accounting for prompt-order sensitivity (Lu et al., 2022). Their mean and standard deviation define the Gaussian expected improvement (Jones et al., 1998), which determines the candidate selected for execution. The candidate count, history limit, sampling procedure, and selection rule are shared across both tasks.

FOREAGENT. We adapt FOREAGENT (Zheng et al., 2026) by generating ten candidate edits and comparing their programs in pairs before execution. The comparisons use the task description, hardware, and a data report prepared once for the search, while candidate generation uses the experiment history. For Autoresearch, the data report describes the training shards and tokenizer; for MLE-bench Lite, it describes the competition's input files. A pairwise comparison eliminates the predicted weaker candidate when confidence reaches 0.7, and uncertain comparisons are repeated. One candidate is then sampled uniformly from the top-ranked group for execution. Both tasks use the same candidate count, confidence threshold, and comparison procedure, with the MLE-bench Lite candidates generated from the parent selected by MLEvolve.

B Additional Experimental Setup

This appendix describes the three ablations in Figure 4. Panel (a) measures how many later proposals are kept, starting from four chosen edits. Panel (b) tests whether the agent picks the right diagnosis and next step, given the recorded predictions, measurements, and scores of an edit. Panel (c) com-

pares searches with and without effect predictions. Training and evaluation follow Appendix A.4. Panels (a) and (c) train each candidate on one A100 GPU. Panel (b) reuses recorded results and trains nothing.

B.1 Effect feedback and proposal retention

Starting cases. We use four starting edits covering changes to attention computation (attention scaling and QKV fusion) and optimization (matrix learning-rate adjustment and momentum cooldown). Each case is named after its starting edit, with Greedy, ScorePred, and EffectPreconception beginning from the same training program and score. Subsequent proposals may change the architecture, optimizer, or training settings, allowing each search to develop beyond the type of edit that defines its starting case.

Search and feedback. For each starting case, we run each method three times with Sonnet 5, proposing one edit at a time until 25 candidates have been trained and scored, excluding failed training runs from this count. Greedy receives the programs and their scores, ScorePred additionally predicts the validation-score change and receives feedback on that forecast, and EffectPreconception receives effect predictions, measurements, and reconciliation as described in Section 3. Because each edit's feedback is available before the next proposal, the methods can develop different candidates and follow different parents after the first step.

Training and evaluation. Each candidate trains from scratch for 1,200 seconds and replaces its parent only if it achieves a lower validation BPB. A tie, a worse score, or a run without a valid score leaves the parent unchanged. For each case and method, we measure proposal retention as $p_r = 100K_r/25$, where K_r is the number of kept candidates among the 25 scored proposals in run r. Figure 4(a) reports the mean of this rate across three independent runs, in percentage points.

B.2 Effect observations and reconciliation

Cases and recorded evidence. We use recorded experiments to compare decisions made from predictions and scores alone with decisions that also receive effect measurements or reconciliation. The analyzed set contains 100 Autoresearch edits from different parents, with 64 from a Sonnet 5 search and 36 from an Opus 5 search, each allowing 1,200 seconds of training per candidate. Each case contains a registered prediction of relative observable change, a free-form prediction with an explicit pass test, probe outputs, and parent and child validation scores. These records supply all evidence for the comparison without additional training.

Feedback conditions. For each recorded edit, we construct three feedback conditions and present them in random order with the same decision prompt. *Prediction plus score* gives the edit description, the effect prediction made before the run, and the parent's and child's validation scores. *Observation* adds the recorded measurements of the effect, but not the verdicts. *Reconciliation* adds the verdicts and whether the edit may get one limited recovery attempt. Within a case, the three versions share the same edit description, prediction, and scores, and the last two share the same raw measurements. Sonnet 5 sees the edit description, not the full parent code. It picks which assumption to revisit, the next step, and whether a recovery attempt is allowed, and explains why.

Reference decisions. We assess these decisions against a reference rule specified before collecting answers. When the predicted effect does not occur, the reference decision is to revise the account of how the edit works. When the effect occurs without a score improvement, it is to examine the expected benefit or an offsetting cost. Unclear evidence calls for improving the measurement, whereas a supported effect and an improved score call for continuing or confirming the result. All three feedback conditions receive the same descriptions of these choices, so their responses can be assessed against a common reference.

Response scoring. Response scoring checks both agreement with this reference and the evidence given for the decision. A script checks whether the diagnosis and next step match the rule exactly. Opus 5 checks whether each explanation rests on the raw evidence and makes no unsupported claim about causes. It sees the three answers in random order, without their version labels. An answer

succeeds only when both choices match the rule and the explanation rests on the evidence. The recovery choice is scored on its own. With prediction and score alone, every answer chooses to fix the measurement, because the measurements are hidden and the prompt says to do this when evidence is missing. This matches the rule only in the three cases with unclear evidence.

Analyzed set and uncertainty. The 100 cases leave out one more case, in which the reconciliation answer failed. It was the first such failure by case number, and we removed it from all three versions. This keeps the cases paired, but it favors reconciliation, because the case was chosen by its result. Over the remaining 100 cases, the success rates are 3.0%, 65.0%, and 74.0%.

B.3 Effect prediction during search

Configurations. Figure 4(c) removes both the registered effect cards and the free-form effect predictions from EffectPreconception. This version keeps the measured history and the outcome relations, so later proposals can still use the evidence of earlier runs; they just make no effect prediction first. Full EffectPreconception records its predictions before each run and returns their verdicts with the measurements. Autoresearch, with plain score feedback, and ScorePred, with direct score prediction, are shown for comparison.

Search and evaluation. Each method runs three independent 100-node searches with Sonnet 5 and 1,200 seconds of training per candidate, using the Autoresearch data and validation procedure in Appendix A.4. Candidates train from scratch, while each search maintains its own best program and experiment history. The node count includes the initial program and all failed or early-stopped runs, with discarded nodes leaving the best BPB unchanged. We record the best validation BPB at nodes 20, 40, 60, 80, and 100 and plot its mean across the three independent searches, with an endpoint label giving the mean at node 100.

C Additional Results and Analysis

C.1 Effect prediction assessments on PostTrainBench

Table 10 reports prediction assessments on AIME 2025 for four PostTrainBench base models. Effects are assessed from free-form predictions and held-out measurements, using the protocol in Appendix A.6.3. For each model, the held rate for effect predictions exceeds that for auxiliary score forecasts, both over all predictions and over assessable predictions.

Table 10: Effect and auxiliary score prediction assessments on AIME 2025. Held uses all predictions as the denominator; Assessable uses only held and refuted predictions. Score forecasts are assessed for observed changes of at least 1.0 point. Bold marks the higher rate for each model.

	Effect prediction		Score prediction	
Base model	Held (%)	Assessable (%)	Held (%)	Assessable (%)
Qwen3-1.7B	**29.2**	**35.0**	16.7	23.5
Qwen3-4B	**45.8**	**50.0**	8.3	9.1
SmolLM3-3B	**29.2**	**31.8**	12.5	13.6
Gemma-3-4B	**29.2**	**31.8**	4.2	11.1

C.2 Comparison with the published Autoresearch H100 run

We compare an EffectPreconception search with Claude Opus 4.6 to Andrej Karpathy's published Autoresearch run (Karpathy, 2026). Both use H100 GPUs and 300 seconds of training per candidate.[1] Within the first 100 nodes, EffectPreconception reaches 0.9674 BPB and the published run reaches 0.9721 (Figure 5). The 100 nodes include the starting program and failed attempts.

[1]Published run: `https://github.com/karpathy/autoresearch/tree/exp/H100/mar8`. The starting BPB is 0.9927 for our run and 0.9979 for the published run.

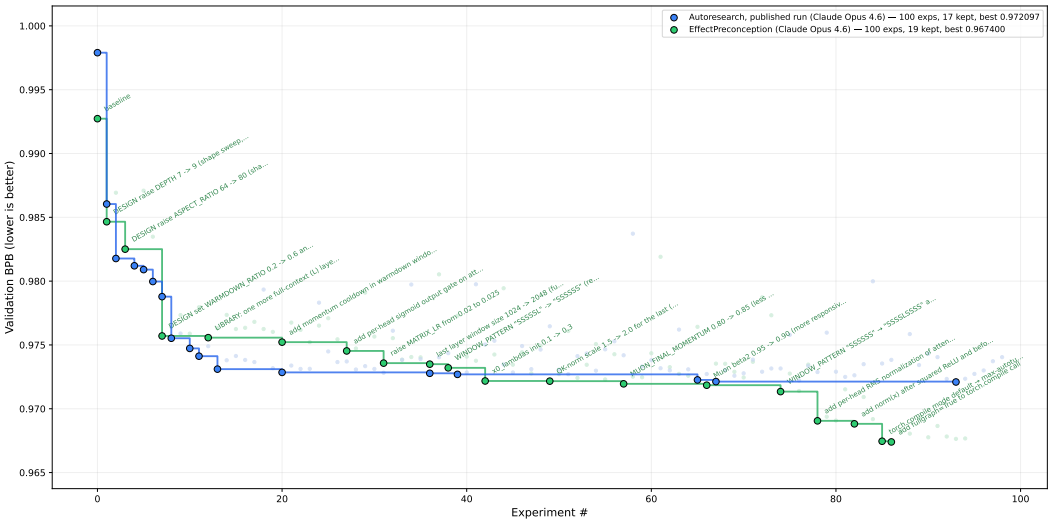


Figure 5: Search progress over the first 100 nodes on one H100 with 300 seconds of training per candidate. Solid circles are kept candidates, faint points are discarded ones within the plotted range, and the lines follow the best BPB so far. The notes describe the kept edits. The horizontal axis counts finished runs from zero; failed attempts count toward the 100 nodes but are not on this axis.

D Computational Cost and Overhead

To evaluate computational efficiency, we compare mean search performance across three Sonnet 5 runs against the auxiliary LLM token consumption across archived 100-node Autoresearch searches using 1,200 seconds of per-candidate training time. This accounting isolates auxiliary usage from total search tokens, tracking LLM calls outside the primary proposer conversation while excluding direct code editing and the supplied ledger summary. Token counts are estimated from raw input and output text at four characters per token, utilizing saved API requests for LLAMBO and FOREAGENT alongside requests reconstructed from prompt templates and execution logs for EffectPreconception and ScorePred. Since ScorePred embeds score forecasts directly within the proposer response, its auxiliary count reflects separate classification and resource-prediction calls, whereas standard Autoresearch requires no auxiliary calls under this accounting scheme.

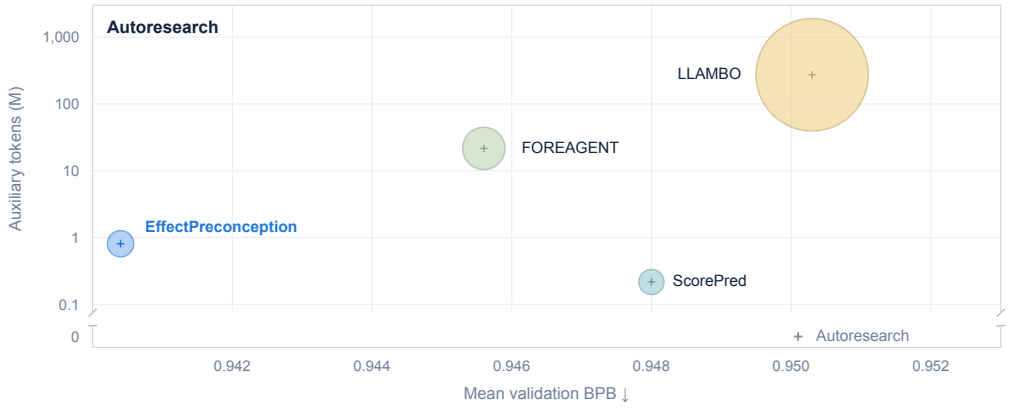


Figure 6: Mean best validation BPB at node 100 across three Sonnet 5 runs (Table 1), plotted against auxiliary token estimates from archived individual searches. Bubble area represents archived auxiliary call attempts. The token axis is logarithmic, with zero shown separately.

EffectPreconception combines lower auxiliary token usage with better mean search performance than LLAMBO and FOREAGENT (Figure 6). It achieves a mean validation BPB of 0.9404, com-

pared with 0.9503 for LLAMBO and 0.9456 for FOREAGENT, while the archived auxiliary token estimates are 0.81, 273.1, and 21.6 million, respectively. ScorePred uses fewer auxiliary tokens (0.22 million) but reaches a higher mean validation BPB of 0.9480. The lower token usage reflects where each method spends its prediction calls. LLAMBO repeatedly samples performance predictions for multiple candidate edits, conditioning each prediction on the task, parent program, and experimental history. FOREAGENT compares candidate programs in pairs and repeats uncertain comparisons. Both procedures therefore process substantial context across multiple calls to select a single candidate for execution, including predictions about candidates that are never trained. EffectPreconception focuses its auxiliary calls on predicting and assessing effects of the executed edit, then preserves the assessments in the evidence ledger for subsequent proposals. Its auxiliary computation thus produces reusable experimental feedback without repeatedly scoring or comparing a pool of competing candidates.

E CASE STUDIES

In this section, we present qualitative case studies illustrating how effect predictions, reconciliation, and the evidence ledger support iterative model search. An attention-gate revision and learned attention scaling connect predicted effects to measured behavior and validation performance, providing concrete examples of the evidence returned by individual experiments. Longer proposal sequences in Autoresearch and PostTrainBench show how this evidence enters subsequent edits, including revisions informed by discarded candidates. A worked ledger example then details how predictions, measurements, and assessments are preserved for later proposals. Throughout, *held* and *refuted* describe effect assessments, while *retained* and *discarded* describe decisions based on validation performance.

E.1 USING EFFECT FEEDBACK TO REFINE ATTENTION GATING

Figure 7 shows how checking an effect prediction provides evidence for a successful revision of an attention-gating design. At node 20 of an Opus 5 search, the agent adds input-dependent gates to attention-head outputs before the output projection, starting from a neutral multiplier of one. Its free-form prediction expects the gates to move away from one and attenuate the final layer more than other layers. In the original search, the prediction holds, with mean multipliers of 0.828 in the final layer and 1.284 elsewhere, although validation BPB worsens from 0.946541 to 0.949158 and the candidate is discarded. A registered card predicting lower final-layer residual RMS is refuted. These measurements identify attenuation in the final layer and amplification elsewhere, giving the revision agent concrete behavior to address. Using this evidence, the revision agent keeps input-dependent gating but restricts the multiplier to attenuation. It also reduces the gate input to 32 features and moves the gate weights from Muon to AdamW.

We then evaluate the parent, original gated candidate, and revised design with seed 42 and the same 1,200-second training budget. The revised gates have mean multipliers of 0.470 in the final layer and 0.706 elsewhere, confirming the predicted ordering while suppressing outputs in both groups. The final-to-middle-layer attention-output RMS ratio is 2.402, exceeding the predicted bound of 2.0. This measurement is taken after the output projection and differs from the residual-stream RMS in the earlier registered card. The feedback therefore records both the confirmed gate pattern and the unmet RMS condition. The revised design recovers the lost performance and improves over the ungated parent. The parent, original gated candidate, and revision reach 0.945781, 0.949081, and 0.945472 BPB, respectively. Across seeds 43–45, the revision improves over the parent by 0.000104–0.001027 BPB and consistently outperforms a variant that applies the revised gate only in the final layer. In this case, effect prediction and reconciliation make a discarded experiment useful by supplying measured behavior that informs a successful follow-up edit.

E.2 LEARNED ATTENTION SCALING

Figure 8 shows the measured effects of learning attention scales at node 37 of the Sonnet 5 search in Appendix E.4. The agent replaces the fixed query multiplier of 1.5 with 54 learned scales, one per head in each of nine layers, while keeping the attention windows unchanged. The free-form prediction expects the scales to move away from their shared starting value and differ across heads.

It also expects a lower mean scale in layer 8, which has the widest attention window. For fixed logits and an attention mask, lowering the scale spreads attention more evenly.

All four conditions of the prediction hold after training. Across the learned scales, the standard deviation is 0.226 and the mean distance from initialization is 0.181, both exceeding the predicted threshold of 0.02. Variation across heads is also evident within layers, with an average standard deviation of 0.207 against a threshold of 0.01. Layer 8 has a lower mean scale than the other layers, 1.473 versus 1.547, matching the predicted ordering. Validation BPB falls from 0.947497 to 0.947018, so the retained candidate provides evidence that the predicted variation across heads and layers occurred alongside the performance improvement.

E.3 Optimization and attention edits

Figure 9 follows the kept path from node 16 to node 25, with two discarded branches. Each edit gets its own predicted effect. Node 18 doubles the embedding learning rate and predicts that the update size will follow the learning-rate schedule; the measured correlation is 0.987. Node 21 adds a momentum cooldown and predicts that momentum falls from 0.95 to 0.85, which the optimizer log shows directly. Node 23 gates the attention output and predicts that the gates will vary across heads and tokens. All three predictions hold, and all three edits are kept.

The two discarded branches leave different evidence despite both worsening BPB. At node 19, weight growth is slower than predicted after weight decay is reduced, so the effect prediction is refuted. At node 24, the recorded probe around the attention-window transition satisfies the predicted smoothness conditions, and the effect is held. Node 25 illustrates the not observable assessment: lowering Adam momentum improves BPB, but the probe returns no measurements. These records keep effect assessment separate from candidate retention.

E.4 Adaptive attention and training changes

Figure 10 follows retained candidates from node 31 to node 39, where improved performance accompanies different effect assessments. At node 32, the agent predicts that learned skip coefficients will grow from near zero, but their mean magnitude starts near 0.500 and falls to 0.135. Validation BPB improves and removing the trained skip connections raises the probe loss, while reconciliation refutes the predicted coefficient trajectory. Nodes 37 and 39 then introduce learned attention scales, related to the scaling used in query–key normalization (Henry et al., 2020), and input-dependent gates. Their free-form predictions of variation across heads, layers, and tokens hold, while their registered loss-component cards are refuted. Together, these edits illustrate assessments tied to specific observables rather than a single verdict on the candidate.

The discarded branches further distinguish the measured patterns behind unsuccessful edits. Node 34 produces stable runtime without the predicted initial slowdown. Node 53 reduces variation in the momentum-buffer magnitude but increases variation in its direction, contrary to the predicted smoothing. At node 50, late weight growth matches the prediction, yet validation BPB does not improve. Reconciliation preserves these differences for subsequent proposals.

E.5 Different hypotheses from a shared parent

Figure 11 shows five candidates from node 90. All five are discarded, but three still produce their predicted effects. Node 91 limits the kernel search to Triton, and the run time becomes stable. Node 92 feeds more input channels to the attention-scaling gate, and the new channels get a real share of the gate's learned weights. Node 93 raises the learning rate of the scalar parameters, and the parameter norm moves mostly early: 90% of its total movement happens in the first 46.9% of training.

The other two candidates do not produce their predicted patterns. Node 95 raises Adam momentum and predicts that the step-to-step loss changes, after removing the trend, will be positively correlated. The measured correlation is negative, early and late in training. Node 99 delays the weight-decay taper and predicts that the weight norms first shrink and then shrink more slowly. Instead the norms grow in every measured phase. The five candidates therefore share the same retention outcome but leave different evidence about their predictions, with three held and two refuted.

E.6 Effect evidence beyond the validation score

Figure 12 contrasts an edit that produces its predicted effect without improving performance with an edit that leaves the implemented computation unchanged. In the first episode, the agent extends earlier attention-gating edits to the MLP output at node 40. The gates depart from their neutral setting and vary across tokens, satisfying the free-form prediction, while validation BPB worsens from 0.9467 to 0.9493. The held assessment establishes that the predicted gate behavior occurred, giving a more specific account of the experiment than the score alone. A value-gate edit at node 41 produces the same combination of a held effect and a worse score. Across the six attention, MLP, and value-gating or scaling edits shown, all free-form predictions hold, four candidates are retained, and two are discarded.

In the second episode, Autoresearch increases Muon's requested Newton–Schulz iteration count from five to six at node 23. The candidate is retained after BPB changes from 0.968314 to 0.968265. However, the implementation iterates over a coefficient list containing only five entries, so increasing the requested count adds no computation. At the next node, the agent identifies this in the code and reverses the change, which is also retained. This episode illustrates the distinction made in Section 3.3: a change in the recorded validation score does not establish that the intended computational effect occurred.

E.7 PostTrainBench episodes

The last two episodes come from the PostTrainBench searches of Table 3. Each is shown next to Autoresearch on the same benchmark and base model. Both start from the same program and have the same budget: 25 experiments with 1,200 seconds of training each (Appendix A.6).

E.7.1 Learning-rate refinement on GPQA Main

Figure 13 follows a learning-rate increase from 2×10^{-5} to 3×10^{-5} on GPQA Main with SmolLM3-3B. The score falls from 32.37 to 28.12 in EffectPreconception and from 30.36 to 28.13 in Autoresearch, and both candidates are discarded. EffectPreconception's held-out measurements show increases in token accuracy and logit margin alongside a wider generalization gap, refuting the prediction that the gap would remain unchanged. The resulting recommendation proposes a smaller increase, and the next candidate uses 2.5×10^{-5} and scores 33.48, the best result in that search. Its three measurements lie between those of the parent and the larger-rate candidate. Autoresearch later lowers the learning rate to 1.5×10^{-5} at node 19, but the candidate scores 29.02 and is discarded. Its best score remains 31.47, achieved at node 13. The saved recommendation and subsequent edit provide a concrete example of effect evidence entering the next proposal.

E.7.2 Zero-score feedback on AIME 2025

Figure 14 follows the beginning of an AIME 2025 search with Qwen3-1.7B, where the initial program scores zero. Three edits increase the number of reasoning traces, shift the data mixture toward OpenR1 math, and extend the sequence length to 4,096 tokens. Each candidate also scores zero, but its effect prediction holds. The measurements record fitting of the reasoning traces and preservation of answer tokens at the end of long examples. The node 5 proposal cites these observations when shifting its focus to answer format and increasing the integer-answer share from 0.70 to 0.95. That candidate solves the first problem, and node 6 reaches 10.00. Autoresearch remains between 0.00 and 6.67 over its 25 experiments. The episode illustrates how measured effects can inform a new proposal when successive validation scores are identical.

E.8 Worked example of an evidence ledger entry

Figure 15 expands node 23 of the search in Appendix E.3 into a complete free-form reconciliation example. The prediction specifies how the learned gates should change, and all five measurement conditions hold after training. The evidence ledger preserves this assessment together with the measured values, improved validation BPB, and retention decision, providing the record described in Section 3.4.

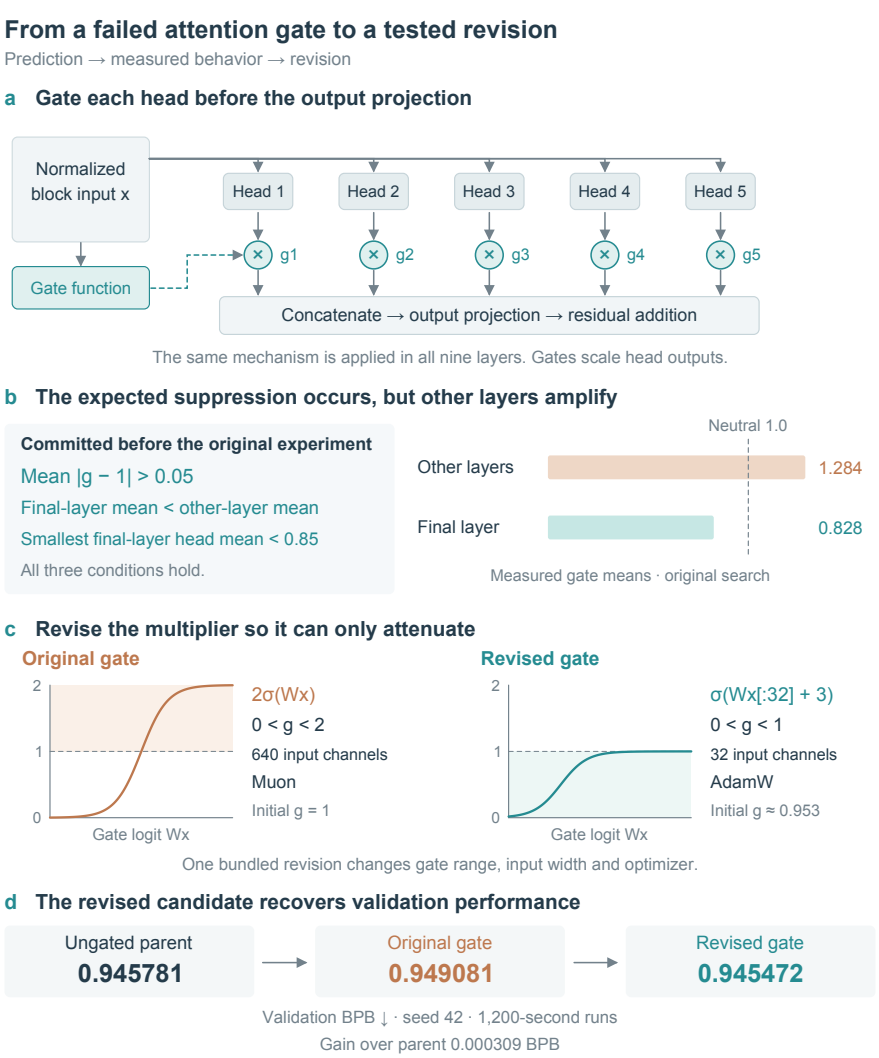


Figure 7: An attention-gate revision informed by the discarded candidate's measurements. Panel (b) reports gate means from the original search. Panel (d) compares the parent, original gated candidate, and revision with seed 42.

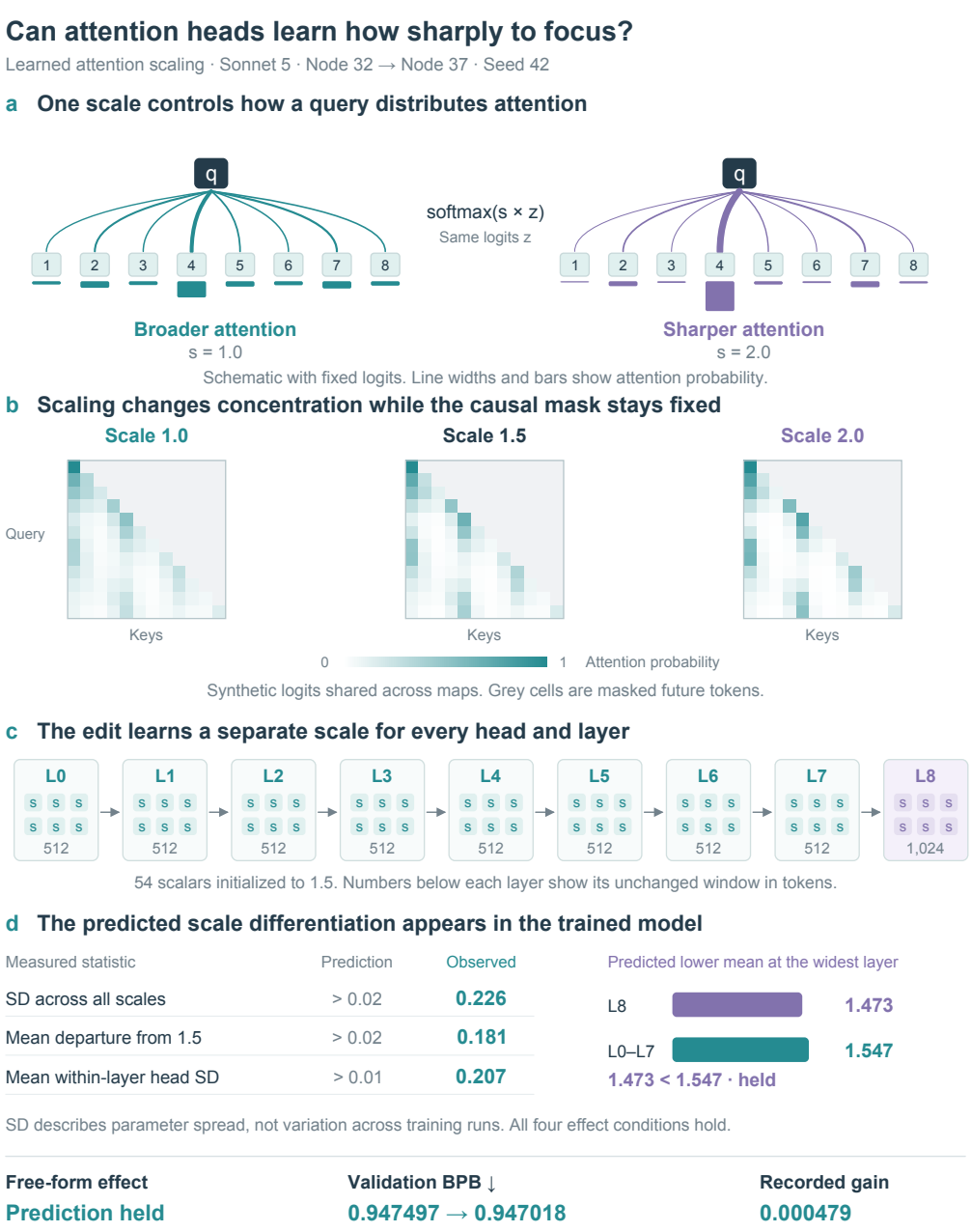


Figure 8: A held free-form prediction for learned attention scales. The first two panels illustrate the effect of scaling on synthetic logits; the remaining panels locate the learned parameters and compare predicted conditions with recorded measurements. Standard deviations describe variation within one trained model.

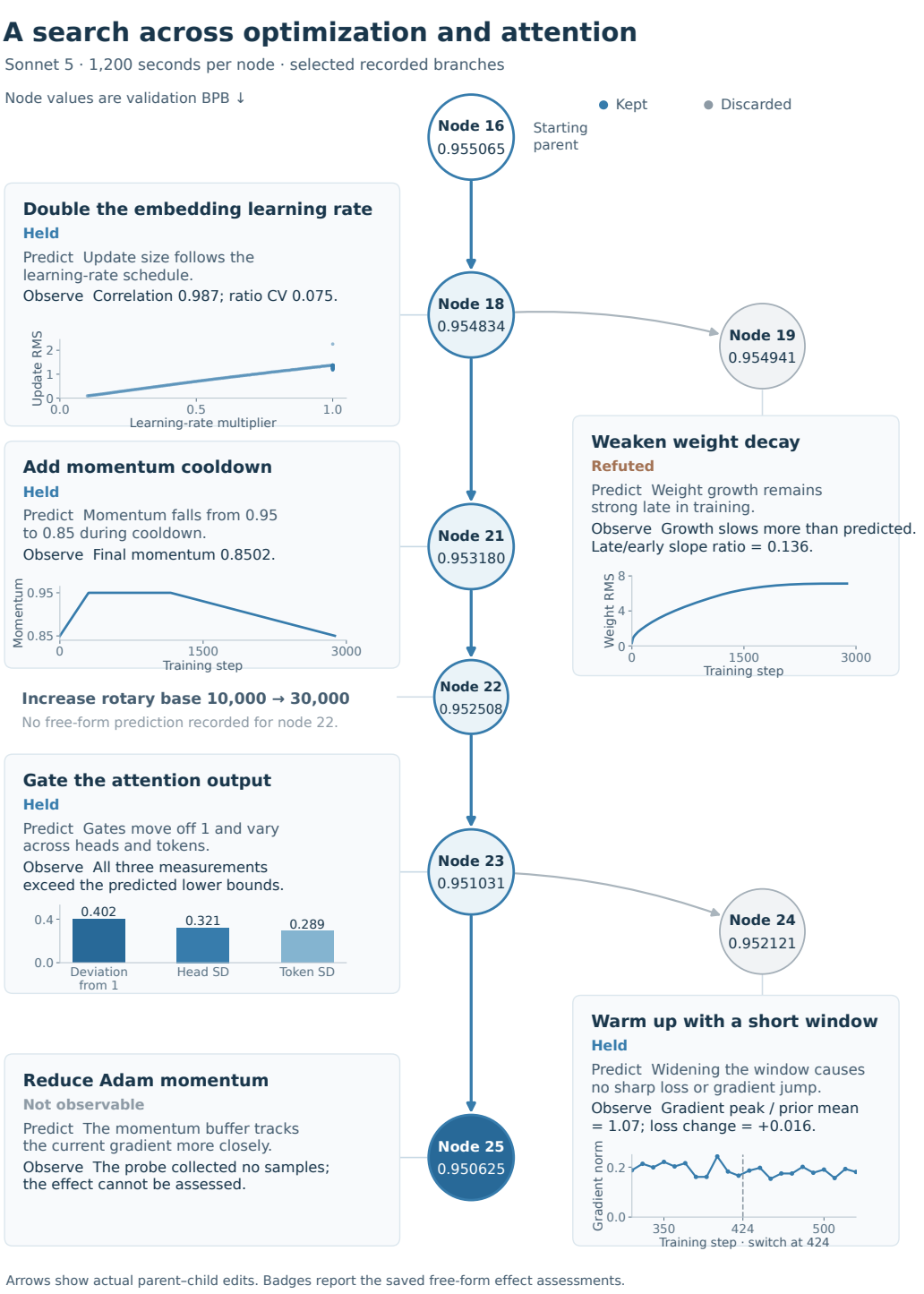


Figure 9: Optimizer and attention edits in one search. The notes show held effects, one refuted prediction about weight growth, and one optimizer measurement that is missing. The effect verdict and the keep or discard decision are shown apart.

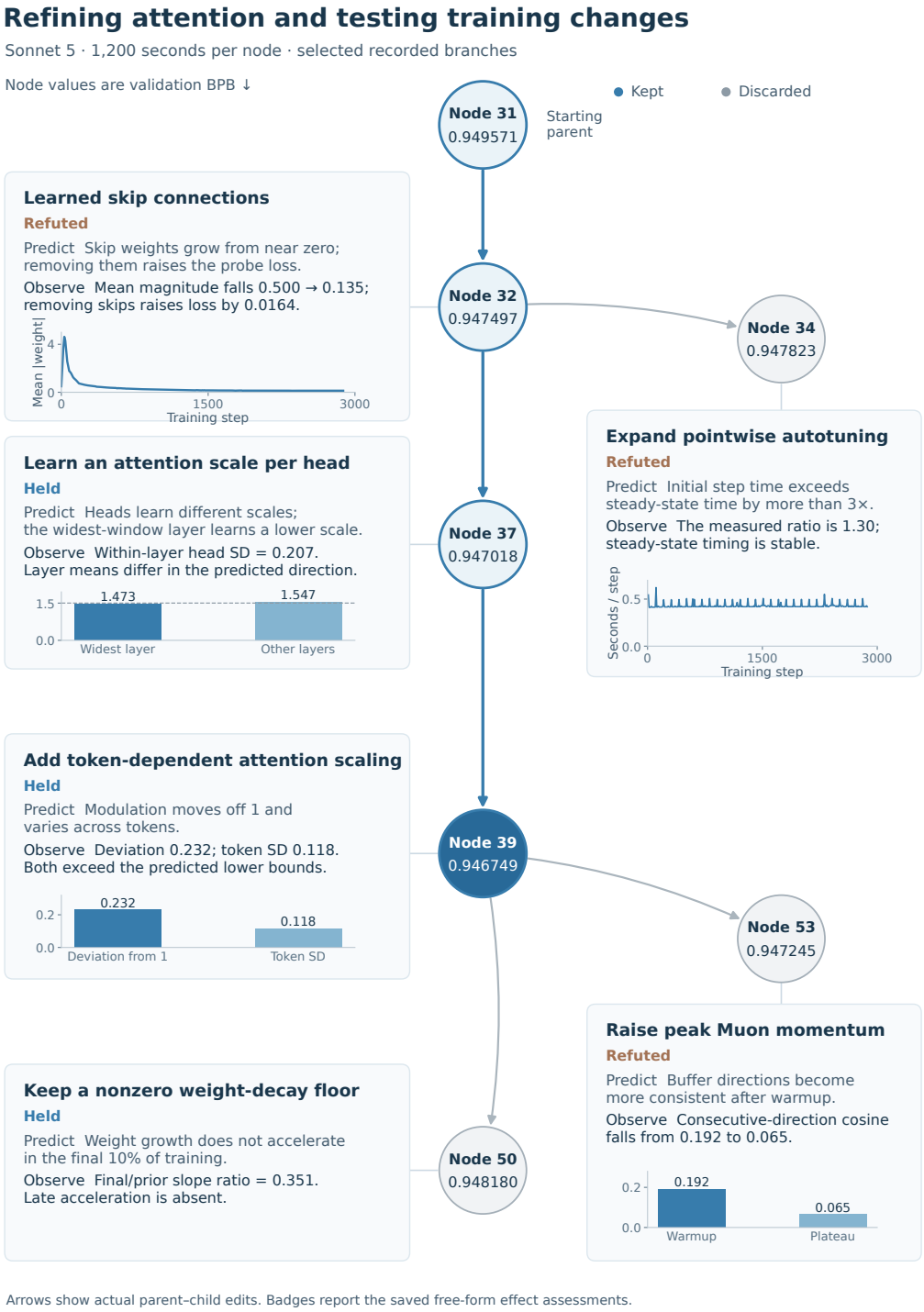


Figure 10: Attention refinements and other training changes. The kept path has both refuted and held effect predictions. The discarded branches test predictions about runtime, weight growth, and the momentum buffer.

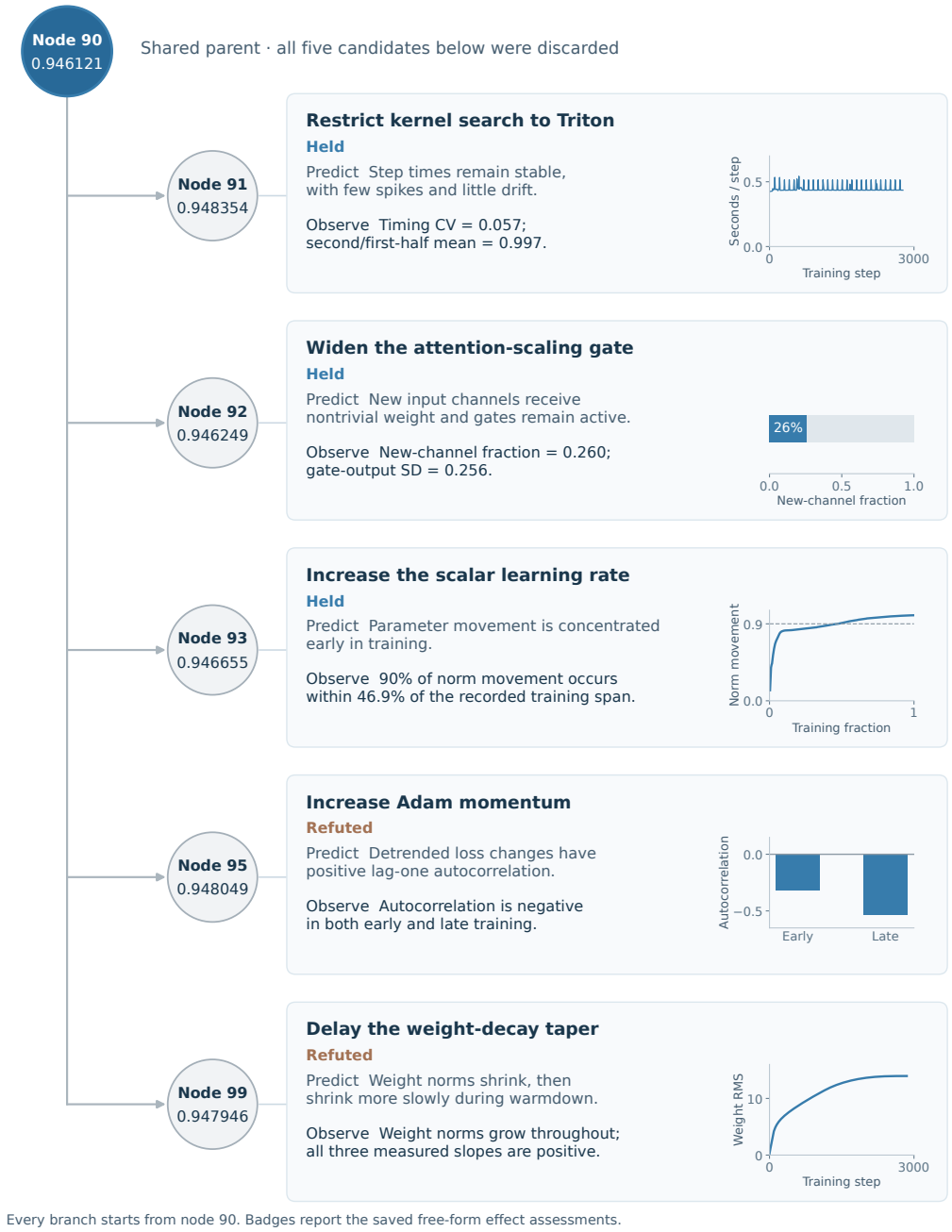


Figure 11: Five discarded candidates from the same parent. Stable run time, use of the new gate inputs, and early movement of the parameter norm match their predictions. Loss autocorrelation and weight growth do not. All five are discarded, but the measurements tell them apart.

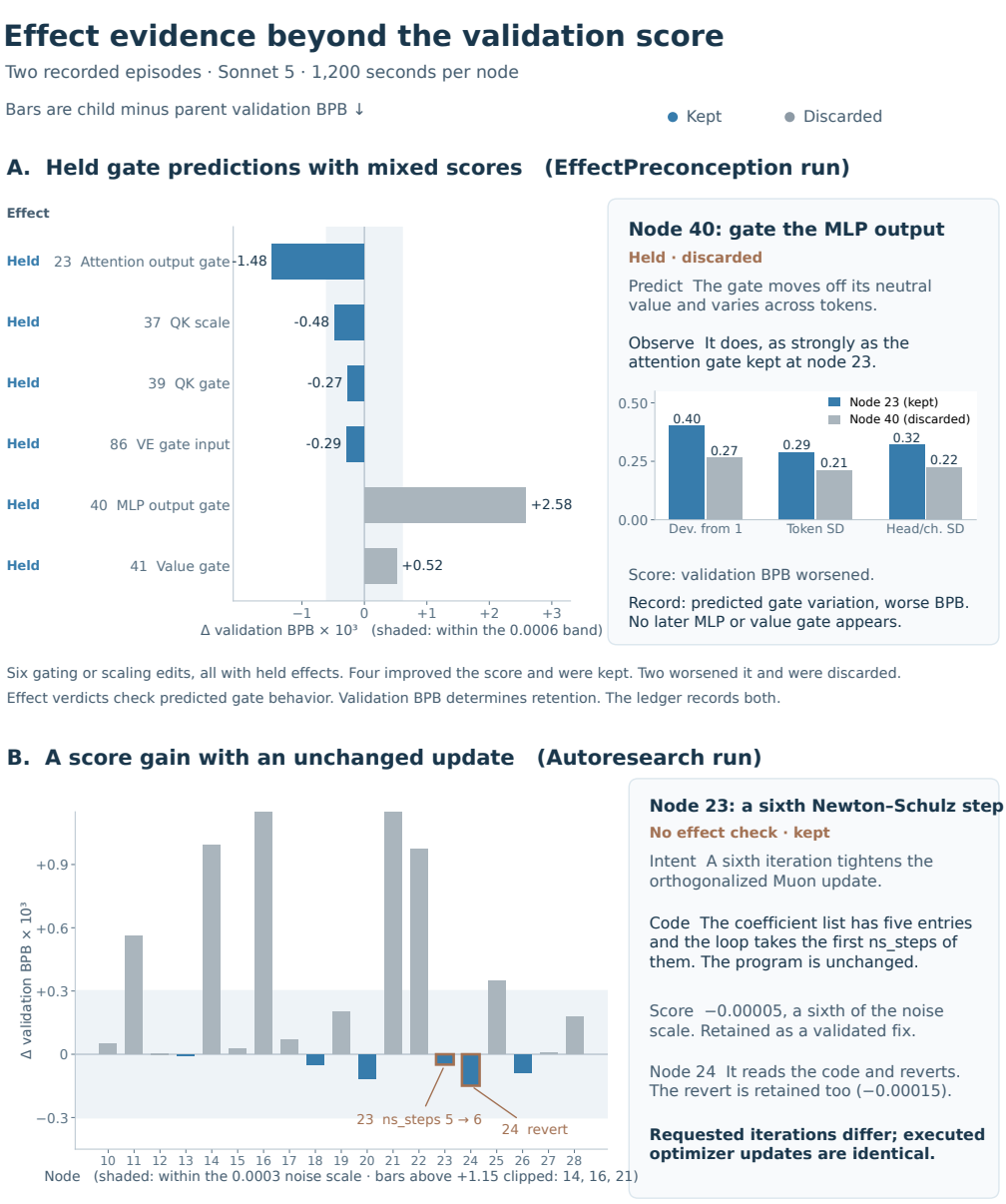


Figure 12: Effect evidence alongside performance evaluation. Top: six gating or scaling edits with held free-form predictions and different retention outcomes. Bottom: increasing the requested Muon iteration count leaves the implemented update unchanged; the candidate and its reversal are both retained after small BPB changes.

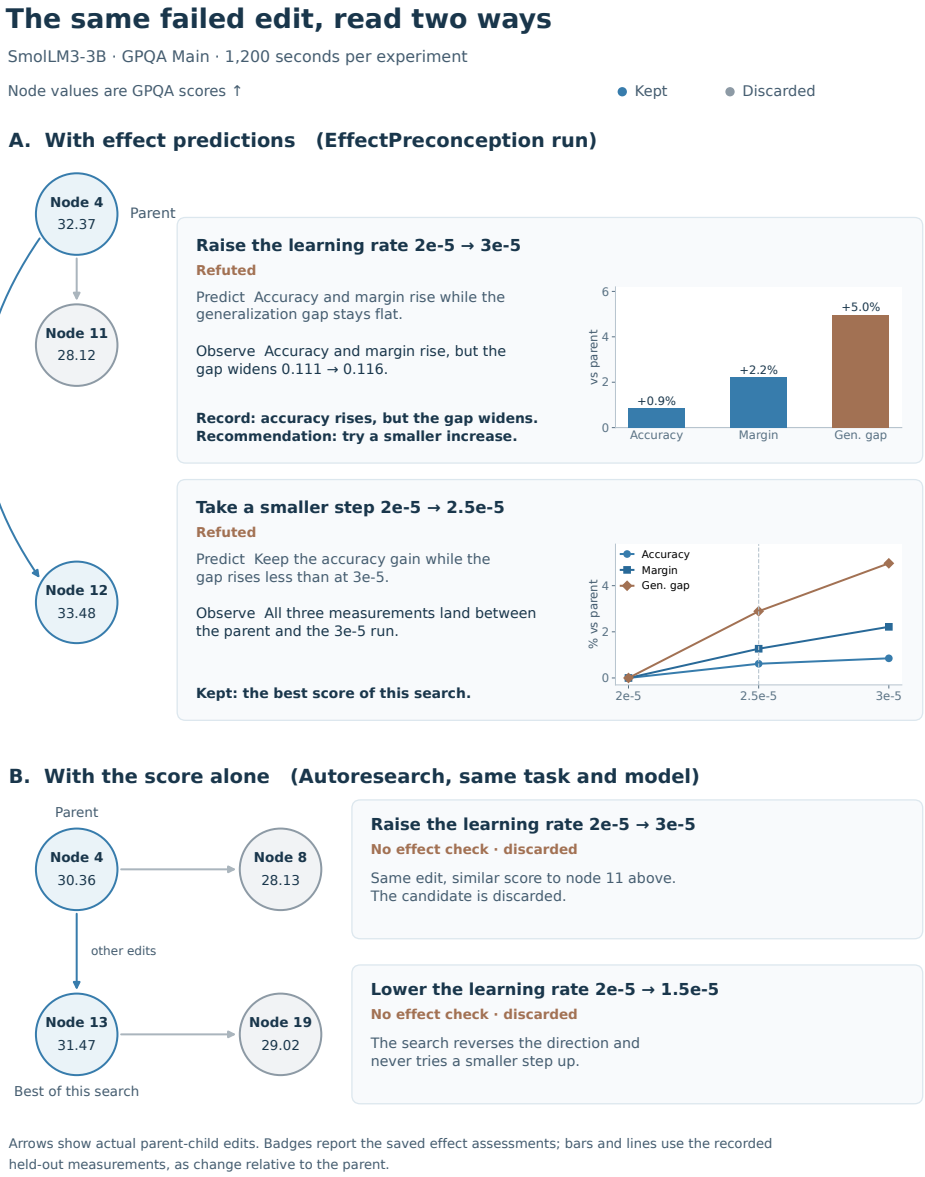


Figure 13: Learning-rate refinement on GPQA Main with SmolLM3-3B. Both searches discard a larger learning-rate increase. EffectPreconception's measurements and refuted prediction lead to a recommendation for a smaller increase, followed by a retained candidate; Autoresearch next decreases the learning rate.

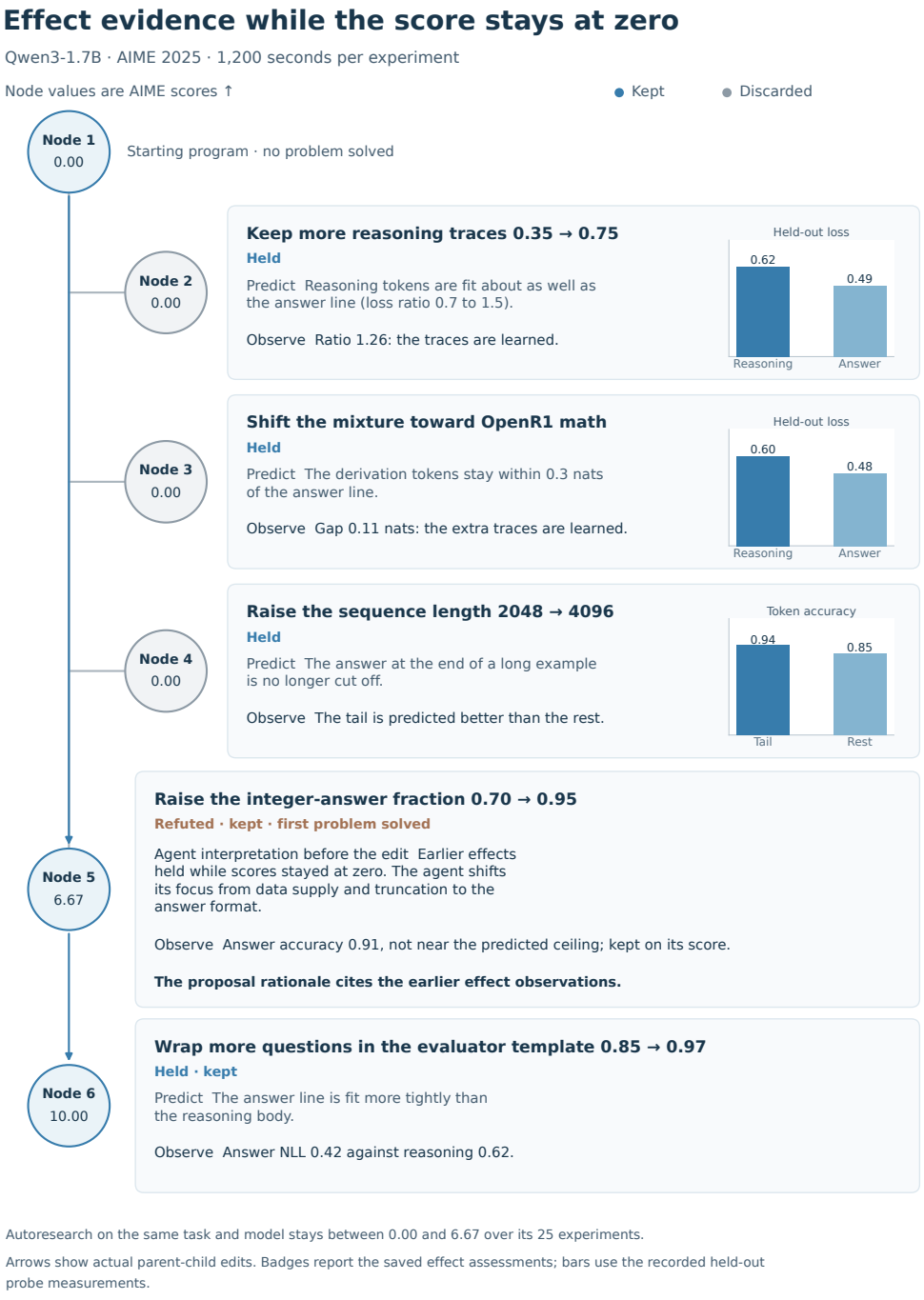


Figure 14: Effect feedback during an initial zero-score plateau on AIME 2025 with Qwen3-1.7B. The first three edits have held effect predictions despite unchanged scores. A subsequent proposal cites their measurements and changes the answer format, producing the first nonzero score.

From effect prediction to evidence ledger

01 Proposed edit

Add an input-dependent gate to each attention head, before the output projection. Zero-initialized weights give a neutral multiplier of 1.

Executed code excerpt

```
out_gate = 2 * torch.sigmoid(self.out_gate(x[..., :self.out_gate_channels]))
y = y * out_gate.unsqueeze(-1)
```

02 Effect prediction

Saved prediction excerpt · omissions marked by […]

"the end-of-training gate values on the held-out batch should have moved meaningfully away from the zero-init neutral value of 1.0 […] AND should vary across heads and positions rather than acting as one global constant"

The recorded conditions also check the final layer, L8.

03 Measurements and reconciliation

Observable	Predicted	Measured	Check
Mean deviation from neutral	> 0.030	**0.402**	**Pass**
Variation across heads	> 0.020	**0.321**	**Pass**
Variation across token positions	> 0.020	**0.289**	**Pass**
L8 deviation from neutral	> 0.020	**0.512**	**Pass**
L8 variation across heads	> 0.015	**0.164**	**Pass**

Effect verdict HELD — Judge and mechanical check agree

04 Evidence ledger entry

Selected saved fields · labels shortened for readability

Record	Node 23 · free-form effect prediction
Parent → candidate	eee4fde → cf889d2
Effect assessment	Held · prediction and measured values preserved
Validation BPB ↓	**0.952508 → 0.951031**
Candidate decision	Retained
Calibration snapshot	Architecture free-form effects held 1 / 1

Figure 15: A saved effect prediction and its resulting ledger entry. Prediction excerpts retain the original wording, with omissions marked; measurements are rounded to three decimals. The ledger preserves confirmation of the predicted effect alongside the improvement in validation BPB.

F LIMITATIONS

EffectPreconception depends on choosing observables that are both measurable and useful for subsequent proposals. Selecting observables that capture an edit's intended effect remains a challenge, particularly when the relevant behavior is difficult to measure directly. More adaptive measurement selection could help agents obtain informative feedback across a broader range of edits. Registered effect cards provide consistent measurement rules for a predefined set of observables, whereas free-form predictions extend this coverage through probes that can themselves contain errors. A useful extension would allow agents to refine their choice of observables as evidence accumulates and verify that each probe measures the intended effect before relying on its feedback.

Even with informative measurements, understanding how effects contribute to performance remains an open problem. Outcome relations connect measured effects to expected changes in validation score, but do not establish causal contributions or whether a relation remains valid as successive edits change the model or training procedure. Extending the framework to design follow-up experiments that distinguish competing explanations could make accumulated evidence more useful across a longer search. Controlled comparisons would also help disentangle the contributions and interactions of effect prediction, outcome relations, and ledger feedback, which our current evaluation does not fully isolate. Together, these directions would support more selective reuse of prior findings, including deciding when an earlier relation needs to be tested again.