Beneath the Diff: Diagnosing and Mitigating Algorithmic Mode Collapse in Code-Level Autonomous Research Loops

Bowei He[1,2], Weixu Zhang[2], Yili Jin[3,4] *, Xue Liu[1,2]
[1] MBZUAI, [2] McGill University, [3] MirrorSpace Technology, [4] Simon Fraser University
Bowei.He@mbzuai.ac.ae, yili@mirrorspace.tech

Abstract

Code-level autonomous research loops (ARLs) have recently emerged as a concrete object of study in automated machine learning research. In such loops, an LLM agent proposes modifications to an experimental training pipeline, executes the modified pipeline, and retains edits that improve a verifiable in-loop metric. Although executable metrics may appear to provide a reliable signal of progress, it remains unclear whether repeated metric-driven code editing leads to genuine improvements that generalize beyond the loop. We provide a systematic diagnosis of this question. Across various experiment settings, we identify a robust failure mode that we call **algorithmic mode collapse**. In this regime, surface-level edit diversity remains stable, but semantic and mechanism-level diversity collapse: the agent continues to edit different lines of code while repeatedly proposing the same kinds of algorithmic changes. This collapse is accompanied by a widening gap between in-loop metric gains and gains measured on independent held-out evaluations. We then propose Diversity-Aware Proposal Sampling (DAPS), a lightweight mitigation that combines category-coverage reweighting, persistent edit memory, and a validation gate. Under a three-tier protocol separating the in-loop metric, the audit metric read by the gate, and a blind metric no loop component ever accesses, DAPS reduces semantic-cluster decay of edits by 69.1% and improves relative faithfulness by 83.7% blind and 81.6% audited, while preserving in-loop optimization speed. We provide the code in Github repository.

1 Introduction

The recent open-source release of iterative autonomous research agents, like the most prominently `autoresearch` by Karpathy (2026) and the broader AI-Scientist family (Lu et al., 2024; Schmidgall et al., 2025; Tang et al., 2026; Gottweis et al., 2025), has made *code-level autonomous research loops* (ARLs) a concrete object of study in automated machine learning research. In such loops, an LLM proposer generates modifications to a training pipeline (optimizer settings, architectural details, data preprocessing, or loss formulations), the modified pipeline is executed, and a validation metric decides whether the edit is retained or reverted. Thus, we can automate the iterative search for better training recipes with minimal human intervention. Unlike recursive training (Kovač et al., 2025) which relies on self-generated training samples to improve the model, code-level ARLs can access optimization signals grounded in an external, executable evaluation. Therefore, one might expect autonomous research loops to be immune to the diversity collapse in recursive training loops, which narrows the space of generated solutions and undermines generalization (Shumailov et al., 2024; Zhu and Xie, 2026; Mishra, 2026).

However, we find that this expectation should be questioned. Code-level ARL, despite using a verifiable external metric, exhibits a previously undocumented failure mode we call **algorithmic mode collapse**: the *surface* of the agent's behavior, such as the lines it edits, the lexical form of its diffs, remains diverse, while the *semantics* of its proposals, what each edit is actually trying to accomplish, progressively concentrates on a small number of recurring patterns. Figure 1 shows the phenomenon at a glance: across four NLP-relevant tasks, surface edit diversity is essentially flat over 500 iterations, while the number of semantic clusters in the proposal stream collapses by 50 to 70%. This narrowing changes the character of the loop. Rather than continuing to search broadly over possible algorithmic interventions, the proposer increasingly returns to a small repertoire of edits that reliably improve the in-loop metric. Such concentration might be benign if it reflected convergence on genuinely useful mechanisms. In our experiments,

*Corresponding to Dr. Yili Jin.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 23652–23669
October 24-29, 2026 ©2026 Association for Computational Linguistics

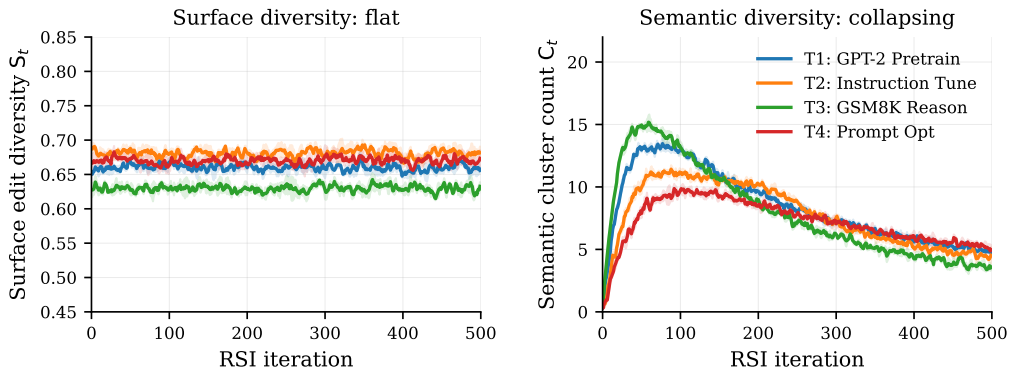

Figure 1: **Algorithmic mode collapse in code-level ARLs.** Left: surface-level edit diversity, measured as mean pairwise normalized edit distance over a sliding window of 20 accepted diffs, remains essentially flat across 500 iterations on all four tasks. Right: semantic-cluster count, obtained by Sentence-BERT (Reimers and Gurevych, 2019) embeddings of LLM-summarized edit descriptions followed by HDBSCAN clustering (Campello et al., 2013), collapses by 50 to 70% over the same horizon. The agent keeps editing different lines of code, but is increasingly proposing the same *kind* of change. Shaded regions are ± 1 standard deviation over 3 seeds.

however, it is accompanied by a growing gap between optimization-metric improvements claimed inside the loop and improvements measured on independent held-out evaluations. As the proposer's repertoire narrows, it increasingly produces edits that overfit the in-loop signal rather than improve the underlying system. By iteration 300, average in-loop gains overstate held-out gains by a factor of 2.0 to 2.6 across our tasks.

Building on this diagnosis, we introduce the mitigation strategy Diversity-Aware Proposal Sampling (DAPS), that injects three lightweight components into an existing ARL loop: a category-coverage reweighting that nudges the proposer toward underrepresented edit mechanisms, a persistent semantic memory that suppresses proposals too similar to recently accepted ones, and a validation gate that reverts the pipeline to its last *faithful* state when the gap between the in-loop metric and a sparingly consulted audit metric exceeds a calibrated threshold. DAPS reduces semantic-cluster decay of edit descriptions by 69.1% and improves the relative faithfulness by 81.6% averaged across tasks, while matching or slightly trailing vanilla autoresearch on in-loop optimization speed. Because the gate consumes the audit metric, we also score every configuration on blind sets that no component ever accesses, where the advantage persists.

Our contributions can be summarized as follows: (i) We conduct the first controlled empirical study of code-level ARL dynamics over long horizons across NLP tasks, producing ARL trajectories with logged proposals, diffs, executions, and held-out evaluations. (ii) We introduce a four-axis diagnostic instrument: surface, semantic, mechanism, and corpus-similarity diversity, together with a faithfulness audit that operationalize *algorithmic mode collapse* and reveal its link to in-loop/held-out divergence. (iii) We propose DAPS, a drop-in mitigation that materially reduces collapse and improves faithfulness across three proposer LLMs without slowing optimization. (iv) We show that surface-only diversity measurements that are common in prior data-level collapse analyses (Li et al., 2026; Mishra, 2026), systematically miss algorithmic mode collapse, motivating semantic-aware monitoring for future ARL deployment.

2 Related Work

Autonomous Research Agents. The AI-Scientist (Lu et al., 2024), Agent Laboratory (Schmidgall et al., 2025), AI-Researcher (Tang et al., 2026), and AI Co-Scientist (Gottweis et al., 2025) systems orchestrate LLM agents through hypothesis generation, experimentation, and writeup. Karpathy (2026) distilled the core experiment-loop pattern into a 630-line script that operates at single-GPU scale, popularizing the term *autoresearch* and motivating a wave of variants (Walker, 2026). Concurrently, Trehan and Chopra (2026) document six recurring failure modes across four end-to-end autonomous ML research attempts, while Xiong et al. (2026) report that even frontier models reach only 9.4% accuracy on rigorous scientific-literature discovery. Our work is complementary: where prior efforts evaluate end-to-end research quality, we analyze the *dynamics* of the experimentation loop.

Diversity Collapse in Recursive Training. Shu-

mailov et al. (2024) established that recursively training on model-generated data leads to distributional narrowing. Subsequent work has refined this picture for self-play and self-training (Kovač et al., 2025; Zhu and Xie, 2026). Most directly related, Li et al. (2026) identify a *Diversity Illusion* in Challenger and Solver self-play. i.e., surface variation persists while underlying patterns collapse. Mishra (2026) diagnose *curriculum collapse* in self-evolving reasoning systems, proposing semantic cluster-coverage rewards. We adopt their semantic-vs-surface framing but transport it to a fundamentally different setting: the optimization signal in code-level ARL is an external execution, not a learned judge or self-generated label, so prior collapse mechanisms (model-on-its-own-data) do not directly apply. Our diagnosis identifies a distinct mechanism rooted in the *proposer's* prior distribution interacting with metric overfitting.

Reward Hacking and Goodharting. That optimizers exploit proxy metrics is a classical observation (Christiano et al., 2017; Skalse et al., 2022; Pan et al., 2022). Closer to our setting, Gao et al. (2023) document that the gap between proxy and gold rewards widens with optimization pressure. Algorithmic mode collapse can be read as a Goodhart-style phenomenon at the *proposal* layer: a narrowing repertoire of edits is selected precisely because it reliably moves the in-loop metric, including when the underlying system has not improved.

Plagiarism of AI-Generated Research. Gupta and Pruthi (2025) show that AI-generated research ideas frequently recapitulate prior work without attribution. We operationalize a related diagnosis, asking whether a proposed modification is essentially retrieved from the proposer's training data.

3 Methodology

3.1 Code-Level ARL: Setup and Notation

A code-level ARL loop is a tuple $(\mathcal{P}, \pi_\theta, \mathcal{E}, R)$ where $\mathcal{P}$ is a training pipeline represented as a set of source files, π_θ is an LLM proposer, $\mathcal{E}$ is an executor that runs a candidate pipeline to obtain a metric value, and $R : \mathbb{R} \to \{0, 1\}$ is an acceptance rule. At iteration t, the proposer samples a textual modification description $d_t \sim \pi_\theta(\cdot \mid \mathcal{P}_t, h_t)$ together with a code patch ρ_t, where h_t summarizes the loop history. The executor evaluates $\mathcal{P}_t \oplus \rho_t$ on an optimization metric m^{opt}; the new pipeline is retained if $R(m^{\text{opt}}(\mathcal{P}_t \oplus \rho_t) - m^{\text{opt}}(\mathcal{P}_t)) = 1$, typically when the metric improves by more than a noise threshold ϵ. Crucially, m^{opt} is what the agent sees; §3.3 defines the two evaluations that it does *not* optimize.

3.2 Multi-Axis Diversity Instrumentation

Existing analyses of diversity collapse measure it primarily through token- or embedding-level signals over generated *text* (Li et al., 2026; Mishra, 2026). In a code-level autonomous research loop, the natural unit is the edit, which admits multiple, partially independent notions of diversity. We instrument four axes as follows.

Surface diversity (S_t). For each accepted edit, we extract its unified diff and compute the normalized Levenshtein distance between every pair of diffs in a sliding window of the $W = 20$ most recent accepted edits. S_t is the window-mean. This captures whether the agent is editing different lines of code in lexically different ways.

Semantic diversity (C_t). The proposer emits, alongside each patch, a one-sentence natural-language description of its intent (or we elicit one post-hoc from a fixed summarizer LLM). We embed all descriptions accumulated up to iteration t with Sentence-BERT (Reimers and Gurevych, 2019) and cluster them with HDBSCAN (Campello et al., 2013) at a fixed minimum cluster size of 5. C_t is the number of clusters whose youngest member was added within the last W iterations. The motivation is direct: *what* the agent is trying to accomplish, like learning-rate adjustment, attention re-formulation, regularization tweak, is exactly what surface metrics fail to capture.

Mechanism entropy (H_t). Surface and semantic axes are continuous; we add a discrete taxonomy for interpretability. To derive the taxonomy we conducted a pilot study on 187 edit descriptions sampled uniformly from publicly available code-level ARL logs, embedded with Sentence-BERT and clustered with affinity propagation (Frey and Dueck, 2007); resulting clusters were iteratively merged when their natural-language summaries overlapped, yielding nine categories that jointly cover 96.4% of pilot edits (a residual "other" bucket absorbs the long tail). Because the pilot logs come from other ARL systems, the categories are frozen before any of our own runs. Table 1 lists the categories with a representative paraphrased example each. At loop time each edit description is assigned by a rule-based parser over a curated keyword inventory, cross-validated against parallel labelling by a frontier LLM annotator; Cohen's κ

Category	Representative edit (paraphrased)
Optimizer	Switch AdamW to Lion; adjust β_2
Scheduling	Lengthen LR warmup; set decay floor
Architecture	Replace LayerNorm with RMSNorm
Data	Filter by sequence length; rebalance mix
Loss	Add label-smoothing or auxiliary term
Regularization	Raise attention-projection dropout
Numerical	FP32 softmax; revise grad-clip bound
Decoding	Adjust top-p / nucleus threshold
Other	Logging, telemetry, build-script edits

Table 1: **Mechanism taxonomy** used for H_t. Categories were derived from a pilot study (see text) and cover 96.4% of observed edits.

between the two on a 200-edit held-out validation set is 0.84, indicating almost perfect agreement (Landis and Koch, 1977). H_t is the Shannon entropy (Shannon, 1948) of the category distribution over a window of W accepted edits. Low H_t in the presence of high S_t is the signature of algorithmic mode collapse: the agent edits diverse lines of code but is increasingly trying to do the same kind of thing. Appendix J shows the collapse to be invariant to taxonomy granularity and to human labelling.

Corpus similarity (V_t). Each edit description is encoded and matched, via nearest-neighbor cosine similarity, against a corpus of 1.1M arXiv cs.CL/cs.LG abstracts published before January 2024. V_t is the mean top-1 similarity over the window. We treat this axis as descriptive: a rising V_t is consistent with the proposer leaning on familiar, frequently described patterns rather than exploring (Gupta and Pruthi, 2025), but it does not by itself establish retrieval, since the same mechanism can be phrased in standard or in unusual ML vocabulary and the corpus is abstracts only, pre-2024, and field-restricted.

3.3 The Three-Tier Faithfulness Audit

We separate three evaluation roles. The *in-loop* metric m^{opt} is the only signal the proposer optimizes and the acceptance rule consumes. The *audit* metric m^{audit}, defined on data the agent cannot infer from $\mathcal{P}_t$ and targeting the *intended* capability rather than the proxy, is never observed by the proposer and is read only by the gate of §3.4 and its threshold calibration, every $K = 10$ iterations, through a scalar test whose only effects are a revert and a templated, value-free notice (2.7 ± 0.9 reverts per run). Since it can thus influence the trajectory, we call it a validation signal rather than a fully held-out test. The *blind* metric m^{blind} is read by no component at any point, gate and calibration included, and is evaluated once per run; it is our headline faithfulness evaluation (partitions in Appendix G).

Every $K = 10$ iterations, we evaluate the best-so-far pipeline on m^{audit} and compare against m^{opt}. Concretely, we define the *faithfulness gap*

$$\Delta_t^{\text{faith}} = \underbrace{\bar{g}_t^{\text{opt}}}_{\text{in-loop gain}} - \underbrace{\bar{g}_t^{\text{audit}}}_{\text{audited gain}}, \qquad (1)$$

where each gain is the improvement over the starting pipeline, signed so that positive means better. A widening Δ_t^{faith} implies that the loop is increasingly optimizing the proxy without commensurate downstream effect; Δ_T^{blind} is defined analogously.

3.4 DAPS: Diversity-Aware Proposal Sampling

To mitigate such identified algorithmic mode collapse, we propose the DAPS framework which composes three components that target the diagnosed pathology directly.

Category-Coverage Reweighting (CCR). Let $p_t(c)$ denote the empirical frequency of mechanism category c in the last W accepted edits. When sampling proposals we draw N candidates from π_θ, classify them on the fly, and re-weight them by $w(\rho) = \exp\big(-\log p_t(c(\rho))/\tau_c\big)$ before acceptance. CCR does not alter the executor; it only changes which candidate is presented for execution. The motivation is that the proposer's prior over edit *types* is sharply peaked toward optimizer/learning-rate edits, and acceptance pressure amplifies skew.

Persistent Edit Memory (PEM). We maintain a first-in-first-out (FIFO) buffer of the Sentence-BERT embeddings of the last $M = 200$ accepted edit descriptions and reject any candidate whose cosine similarity to its nearest neighbor in memory exceeds τ_m. PEM prevents the loop from cycling on near-duplicate semantics regardless of whether the underlying code differs. This is the code-level analog of the memory-augmented penalty of Li et al. (2026), applied to edit descriptions rather than self-play questions.

Audit-Based Validation Gate (HVG). Every $K = 10$ iterations we compute Δ_t^{faith} from Eq. 1. If Δ_t^{faith} exceeds a task-calibrated threshold τ_h, the gate reverts $\mathcal{P}$ to the last checkpoint with $\Delta^{\text{faith}} \leq \tau_h$ and feeds a structured failure summary back to the proposer. HVG is the only component that consumes the audit metric; it does so sparingly, through the scalar test of §3.3, and is the mecha-

Algorithm 1 Diversity-Aware Proposal Sampling

Input: pipeline $\mathcal{P}_0$, proposer π_θ, executor $\mathcal{E}$, metrics $m^{\text{opt}}, m^{\text{audit}}$
Initialize memory $\mathcal{M} \leftarrow \emptyset$, history $h \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  Sample N candidates $\{(\rho_i, d_i)\} \sim \pi_\theta(\cdot \mid \mathcal{P}_{t-1}, h)$
  Compute $w_i \leftarrow \exp(-\log p_t(c(\rho_i))/\tau_c)$ ▷ CCR
  Drop ρ_i if $\max_{e \in \mathcal{M}} \cos(\phi(d_i), e) > \tau_m$ ▷ PEM
  Choose $\rho^\star \leftarrow$ weighted-sample remaining candidates by w_i
  $m \leftarrow \mathcal{E}(\mathcal{P}_{t-1} \oplus \rho^\star)$
  if $R(m - m^{\text{opt}}(\mathcal{P}_{t-1})) = 1$ **then**
    $\mathcal{P}_t \leftarrow \mathcal{P}_{t-1} \oplus \rho^\star$; add $\phi(d^\star)$ to $\mathcal{M}$
  else
    $\mathcal{P}_t \leftarrow \mathcal{P}_{t-1}$
  end if
  if $t \bmod K = 0$ **and** $\Delta_t^{\text{faith}} > \tau_h$ **then** ▷ HVG
    Revert $\mathcal{P}_t$ to last faithful checkpoint
    Append failure summary to h
  end if
end for

nism by which Goodhart-style edits are eventually undone.

The full procedure has been summarized in Algorithm 1.

4 Experiments

4.1 Experiment Setup

4.1.1 Tasks and Datasets

We instantiate code-level ARL on four NLP-relevant tasks spanning pretraining, post-training, reasoning, and inference-time configuration.

T1: Small-LM Pretraining. A GPT-2-small style model (Radford et al., 2019) (∼124M parameters) trained on a 0.5B-token subset of OpenWebText for a fixed 5-minute budget per execution, following the `autoresearch` setup of Karpathy (2026). m^{opt}: in-distribution validation loss. m^{audit}: perplexity on LAMBADA (Paperno et al., 2016) and a held-out C4 (Raffel et al., 2020) subset.

T2: Instruction Tuning. A Llama-3.2-1B base model (Grattafiori et al., 2024) fine-tuned on a 20k-example Alpaca subset (Taori et al., 2023) via LoRA. m^{opt}: held-in instruction-following win-rate (AlpacaEval-style (Li et al., 2023) against a fixed reference) on a 500-example development split. m^{audit}: MMLU (Hendrycks et al., 2021a) 5-shot accuracy and the IFEval prompt-following benchmark (Zhou et al., 2023).

T3: Reasoning Fine-Tuning. A Qwen-2.5-1.5B (Yang et al., 2024) model fine-tuned on GSM8K (Cobbe et al., 2021) training split. m^{opt}: GSM8K dev-set exact-match accuracy. m^{audit}: ARC-Easy (Clark et al., 2018) and MATH-500 (Hendrycks et al., 2021b) subset accuracy.

T4: Prompt Optimization. Frozen Llama-3.2-3B prompted on a question-answering subset, where the autonomous research loop edits a Python DSPy-style prompt program (Khattab et al., 2023). m^{opt}: dev-set accuracy on a sampled ARC-Easy split. m^{audit}: ARC-Challenge (Clark et al., 2018) and CommonsenseQA (Talmor et al., 2019).

The m^{opt} and m^{audit} are computed with independent prompts and data partitions. Here, audit tasks are chosen to be *capability-overlapping but distribution-different* relative to in-loop ones: a genuine capability improvement should transfer, while a metric-specific overfitting should not. The blind sets, evaluated once per run and read by no component of the loop, are WikiText-103 log-perplexity (Merity et al., 2017) for T1, win-rate on 500 held-out Dolly instructions (Conover et al., 2023) under the same judging protocol for T2, SVAMP accuracy (Patel et al., 2021) for T3, and OpenBookQA accuracy (Mihaylov et al., 2018) for T4.

4.1.2 Baselines

We compare DAPS against eight baselines spanning the major existing strategies. (**B1**) VANILLA AR: a faithful reimplementation of Karpathy (2026) with the same proposer LLM. (**B2**) HITEMP: vanilla autoresearch with proposer temperature raised from 0.7 to 1.2, a frequently suggested ad-hoc fix for low diversity. (**B3**) R-DIVERSE-A: vanilla autoresearch augmented with the Memory-Augmented Penalty of Li et al. (2026), applied to edit descriptions (the strongest published mitigation transposed to our setting). (**B4**) PRISM-A: vanilla autoresearch with the semantic cluster-coverage reward of Mishra (2026). (**B5**) REFLEXION: vanilla autoresearch with an additional reflective summary (Shinn et al., 2023) prepended to h at every step. (**B6**) RANDSEARCH: a proposer-free baseline that samples edits uniformly from a corpus of code modifications harvested from the proposer LLM in iteration 1; this isolates the contribution of LLM's adaptive proposals beyond a static edit

distribution. (**B7**) HO-EARLYSTOP: vanilla autoresearch that reads m^{audit} every $K{=}10$ iterations and returns the best-audit checkpoint. (**B8**) HO-REVERT: audit-based reversion alone, that is HVG without CCR or PEM. B7 and B8 match DAPS in audit frequency, compute, and feedback format, while HITEMP and RANDSEARCH are diagnostic controls rather than competitors. All baselines share DAPS's tuning budget (Appendix N).

4.1.3 Evaluation Metrics

For each method/task/seed trajectory, we report: (i) *In-loop gain* $\bar{g}_T^{\text{opt}}$; (ii) *Audited gain* $\bar{g}_T^{\text{audit}}$ and *blind gain* $\bar{g}_T^{\text{blind}}$; (iii) *Faithfulness ratios* $\rho_T^{\text{audit}} = \bar{g}_T^{\text{audit}}/\bar{g}_T^{\text{opt}}$ and $\rho_T^{\text{blind}} = \bar{g}_T^{\text{blind}}/\bar{g}_T^{\text{opt}}$ (closer to 1 is better; < 1 indicates Goodharting), where all gains are signed improvements oriented so that positive means better: on T1 numerator and denominator are both reductions in nats per token ($0.058/0.142 = 0.41$ under VANILLA AR), on T2 to T4 both are percentage points, and ratios are reported only when $\bar{g}_T^{\text{opt}} > \epsilon$, as held throughout. We write ρ_T without a superscript when a statement holds for both; (iv) *Faithfulness gap* Δ_T^{faith} at end of run (Eq. 1); (v) The four diagnostic axes of §3.2 evaluated at end of run: *surface diversity* S_T, *semantic cluster count* C_T, *mechanism entropy* H_T, and *corpus similarity* V_T; (vi) *Cluster decay* $\Delta\mathrm{C} = (\mathrm{C}_{T/5} - \mathrm{C}_T)/\mathrm{C}_{T/5}$, the fractional drop in clusters from the early-warm-up to the final window, used as a scalar summary of trajectory-level collapse. We report mean $\pm$ s.d. over 3 seeds for the main configurations and aggregate 5 seeds for T1 (where per-trajectory variance is largest).

4.1.4 Implementation Details

In experiments, our primary pipeline forks the public `autoresearch` repository (Karpathy, 2026) and adds task adapters, the four diagnostic axes, and the DAPS components. To verify that our findings are not artifacts of a single implementation, we additionally instantiate the same ARL specification on top of Aider (Gauthier, 2024), a popular open-source code-editing agent whose proposal mechanism differs substantially from `autoresearch`: Aider emits structured `SEARCH/REPLACE` blocks committed via git rather than direct unified diffs, manages its own multi-file repository map and chat history, and decouples "decide what to change" from "apply the change" through separate LLM passes. We wrap Aider as a drop-in proposer-and-editor module within our loop while keeping the executor, acceptance rule, history summary, and diagnostic instrumentation identical. The diagnostic axes are computed in exactly the same way on both frameworks: edit descriptions are extracted from Aider's commit messages (which it auto-generates) and re-summarized by the same fixed summarizer LLM for consistency with the `autoresearch` pipeline. The proposer LLM for our main configuration is Claude Opus 4.7 with temperature$=0.7$, sampled through the Anthropic API; robustness ablations additionally use GPT-5.2 and Llama-3.3-70B served via vLLM. For semantic embedding we use `all-mpnet-base-v2` (Reimers and Gurevych, 2019); clustering uses HDBSCAN with min_cluster_size $= 5$ and min_samples $= 2$ (Campello et al., 2013); UMAP (McInnes et al., 2018) is used only for visualization. Hyperparameters of DAPS are fixed across all tasks and both frameworks at $\tau_c{=}1.0$, $\tau_m{=}0.85$, $M{=}200$, $K{=}10$, and a task-calibrated τ_h chosen on the first 30 iterations to be the 80th percentile of $|\Delta^{\text{faith}}|$ observed under VANILLA AR. Each ARL execution is bounded at 5 minutes on a single A100; full sweeps used approximately 6,200 A100-hours.

4.2 Main Results and Analysis

We report our main results in Table 2. Three findings stand out. *First, vanilla autoresearch overfits.* Across all four tasks, ρ_T^{audit} for VANILLA AR ranges from 0.38 to 0.49, meaning roughly half to two-thirds of the gain claimed inside the loop fails to materialize on held-out evaluation. This is the quantitative form of the gap previewed in §1. *Second, data-level diversity mitigations transfer only partially.* R-DIVERSE-A and PRISM-A, the strongest published collapse mitigations from the data-level literature, improve faithfulness (ρ_T^{audit} rises from ≈ 0.43 to ≈ 0.58 averaged across tasks) but neither fully closes the gap nor preserves in-loop performance ($\bar{g}^{\text{opt}}$ degrades by 3% to 13% relative to VANILLA AR on T2 and T3). This supports our claim that code-level ARL involves a distinct failure mode requiring code-level instrumentation. *Third, raising temperature is not a remedy.* HITEMP achieves marginally higher faithfulness at marginally lower in-loop gain, but the effect is well within the standard deviation. Increased proposer entropy does not, in our experiments, translate to increased *semantic* diversity, a phenomenon we examine in §4.4. DAPS achieves the highest ρ_T^{audit} on all four tasks while remaining within one standard deviation of the best $\bar{g}^{\text{opt}}$. We attribute the absence

Method	T1 (Pretrain)		T2 (InstrTune)		T3 (Reason)		T4 (Prompt)	
	$\bar{g}^{\text{opt}}$	ρ_T^{audit}	$\bar{g}^{\text{opt}}$	ρ_T^{audit}	$\bar{g}^{\text{opt}}$	ρ_T^{audit}	$\bar{g}^{\text{opt}}$	ρ_T^{audit}
VANILLA AR	$+0.142_{\pm.011}$	$0.41_{\pm.05}$	$+8.9_{\pm1.2}$	$0.46_{\pm.07}$	$+11.7_{\pm1.8}$	$0.38_{\pm.06}$	$+9.4_{\pm1.1}$	$0.49_{\pm.05}$
HITEMP	$+0.131_{\pm.014}$	$0.43_{\pm.06}$	$+8.4_{\pm1.4}$	$0.49_{\pm.06}$	$+10.9_{\pm2.1}$	$0.40_{\pm.08}$	$+9.0_{\pm1.3}$	$0.51_{\pm.06}$
R-DIVERSE-A	$+0.128_{\pm.012}$	$0.55_{\pm.05}$	$+8.1_{\pm1.0}$	$0.58_{\pm.06}$	$+10.2_{\pm1.6}$	$0.51_{\pm.07}$	$+8.7_{\pm1.2}$	$0.60_{\pm.05}$
PRISM-A	$+0.135_{\pm.010}$	$0.59_{\pm.04}$	$+8.6_{\pm1.1}$	$0.61_{\pm.05}$	$+10.8_{\pm1.5}$	$0.54_{\pm.06}$	$+9.1_{\pm1.0}$	$0.62_{\pm.04}$
REFLEXION	$+0.138_{\pm.013}$	$0.47_{\pm.06}$	$+8.7_{\pm1.3}$	$0.50_{\pm.07}$	$+11.2_{\pm1.7}$	$0.43_{\pm.07}$	$+9.3_{\pm1.1}$	$0.53_{\pm.05}$
RANDSEARCH	$+0.073_{\pm.018}$	$0.78_{\pm.09}$	$+4.1_{\pm1.5}$	$0.81_{\pm.10}$	$+5.6_{\pm1.9}$	$0.74_{\pm.11}$	$+5.0_{\pm1.4}$	$0.79_{\pm.09}$
DAPS (ours)	$\mathbf{+0.139_{\pm.009}}$	$\mathbf{0.78_{\pm.04}}$	$\mathbf{+8.8_{\pm1.0}}$	$\mathbf{0.80_{\pm.05}}$	$\mathbf{+11.5_{\pm1.4}}$	$\mathbf{0.74_{\pm.06}}$	$\mathbf{+9.3_{\pm0.9}}$	$\mathbf{0.82_{\pm.04}}$

Table 2: **Main results.** In-loop gain $\bar{g}^{\text{opt}}$ and audited faithfulness ratio ρ_T^{audit} at $T = 300$ across four tasks. Both are oriented so that higher is better: $\bar{g}^{\text{opt}}$ is the loss reduction in nats per token on T1 and the win-rate or accuracy gain in percentage points on T2 to T4. ρ_T^{audit} closer to 1 indicates that in-loop gains transfer to the audit evaluation; blind-set counterparts are in Table 3 and Appendix G. Subscripts denote ±1 standard deviation over 3 seeds (5 for T1).

Method	audit	$\bar{g}^{\text{opt}}$	ρ_T^{aud}	ρ_T^{bli}	ΔC
VANILLA AR	none	1.000	0.44	0.41	0.68
R-DIVERSE-A	none	0.900	0.56	0.52	0.39
PRISM-A	none	0.950	0.59	0.55	0.34
CCR+PEM	none	0.986	0.71	0.67	0.28
HO-EARLYSTOP	K=10	0.910	0.62	0.57	0.66
HO-REVERT	K=10	0.953	0.67	0.62	0.55
DAPS (full)	K=10	0.983	**0.79**	**0.74**	**0.21**

Table 3: Faithfulness under matched audit access at $T = 300$, averaged across tasks and seeds, with $\bar{g}^{\text{opt}}$ normalized by VANILLA AR and ρ^{aud}, ρ^{bli} abbreviating ρ^{audit}, ρ^{blind}. Methods in the upper block never read any external metric; those in the lower block read m^{audit} every K iterations under the same evaluation frequency, compute, and feedback format.

of an in-loop cost to two facts: CCR/PEM reject duplicate proposals *before* executor calls, so they do not waste budget; and HVG reverts only when in-loop and audit metrics diverge, so a faithful trajectory is unaffected.

4.3 Faithfulness Under Matched Audit Access

Table 3 studies whether the faithfulness advantage survives on data the gate never touches, and whether it follows from the method or from audit access that no baseline is granted. Every configuration loses only 0.03 to 0.05 between the audit and the blind tier, including the four that read no external metric at all, so that offset reflects benchmark idiosyncrasy rather than leakage through the gate: DAPS keeps its advantage blind (0.74 against 0.79 audited), and per task the relative improvement over VANILLA AR is 83.7% blind against 81.6% audited. On equal footing, three conclusions follow. *First*, with zero audit access CCR+PEM already surpasses both published diversity mitigations on both faithfulness tiers and on ΔC at a smaller in-loop cost. *Second*, audit access alone reaches at most $\rho_T^{\text{audit}} = 0.67$ (0.62 blind) and barely reduces collapse ($\Delta\text{C} \geq 0.55$), so the extra signal explains only part of the advantage. *Third*, the two ingredients are complementary, as §3.4 intends. Appendix F indexes the validity, robustness, and scope checks behind these numbers.

4.4 Dissecting the Collapse

Figure 2 traces our four diagnostic axes on T2 under VANILLA AR and DAPS. We have three observations as follows. (a) *Surface diversity is a poor warning sign.* S_t stays in $[0.66, 0.68]$ for both methods throughout, while H_t under VANILLA AR drops from 1.76 nats at $t = 30$ to 1.18 nats at $t = 300$. Any monitor that watched only S_t would have raised no flag. (b) *Corpus similarity rises with collapse.* V_t increases from 0.60 to 0.72 under VANILLA AR, so the agent concentrates on edits whose descriptions are increasingly close to commonly published ML text (Gupta and Pruthi, 2025). This is consistent with the proposer relying on its prior rather than exploring, although as a descriptive statistic it does not on its own establish retrieval (§3.2). (c) *Faithfulness erodes monotonically.* Δ_t^{faith} grows from ≈ 0.05 at $t = 60$ to ≈ 0.17 at $t = 300$ under VANILLA AR, mirroring the entropy collapse with almost no iteration lag. Under DAPS, the gap stabilizes at ≈ 0.07. Table 4 reports the axis statistics across all tasks and methods. The pattern is consistent: methods that close the entropy gap (PRISM-A, DAPS) also close the faithfulness gap; methods that do not (HITEMP, REFLEXION) do not. Surface diversity correlates with neither.

4.5 Ablation Study

Table 5 ablates the three DAPS components. CCR and PEM attack semantic concentration directly and together reduce ΔC from 0.68 to 0.28. HVG

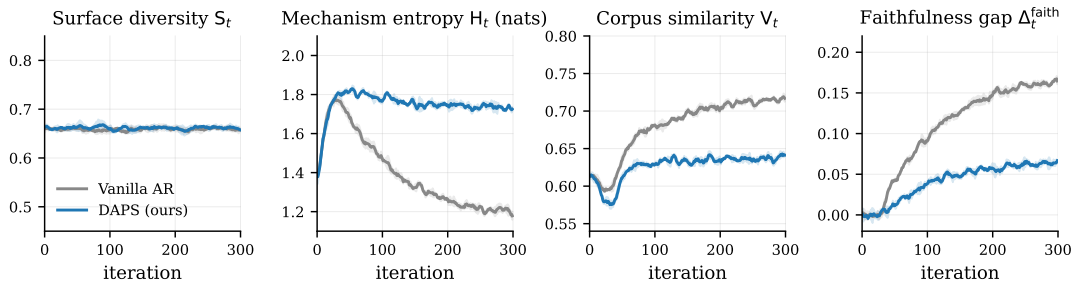

Figure 2: Diagnostic axes over iterations on T2. Mechanism entropy H_t collapses by $\sim$0.6 nats while surface diversity S_t remains flat. Corpus similarity V_t rises, which is consistent with reliance on patterns close to pretraining text; the in-loop/audit gap Δ_t^{faith} widens monotonically under VANILLA AR and is materially shrunk under DAPS.

Method	S_T	H_T	ΔC	Δ_T^{faith}
VANILLA AR	0.66	1.18	0.68	0.171
HITEMP	0.69	1.24	0.61	0.158
R-DIVERSE-A	0.67	1.51	0.39	0.108
PRISM-A	0.65	1.59	0.34	0.094
REFLEXION	0.67	1.27	0.59	0.149
DAPS	**0.66**	**1.71**	**0.21**	**0.067**

Table 4: Diagnostic axes at $T = 300$, averaged across tasks and seeds. Surface diversity S_T is essentially constant across methods; mechanism entropy H_T, cluster decay ΔC, and faithfulness gap Δ_T^{faith} co-vary tightly.

Variant	$\bar{g}^{\text{opt}}$ (norm.)	ρ_T^{audit}	ΔC
VANILLA AR	1.000	0.44	0.68
+ CCR only	0.992	0.58	0.46
+ PEM only	0.978	0.61	0.39
+ HVG only	0.953	0.67	0.55
+ CCR + PEM	0.986	0.71	0.28
+ CCR + HVG	0.969	0.74	0.42
+ PEM + HVG	0.961	0.75	0.31
DAPS (full)	0.983	**0.79**	**0.21**

Table 5: Ablation study of DAPS components averaged across tasks. $\bar{g}^{\text{opt}}$ is normalized by VANILLA AR. Each component contributes; HVG contributes most to faithfulness, CCR+PEM contribute most to cluster decay.

attacks Goodharting and yields the single largest jump in ρ_T^{audit} ($0.44 \rightarrow 0.67$); however, used alone it leaves cluster decay almost untouched ($0.68 \rightarrow 0.55$), because the proposer's prior is not modified. The three contributions are sub-additive: alone they raise ρ_T^{audit} by 0.14, 0.17, and 0.23, which would predict 0.98 rather than the observed 0.79, with the largest shortfall for the pairs that include HVG. The same overlap shows in the in-loop column, where HVG alone is the costliest variant (0.953) while the full system recovers most of that cost, consistent with a broader proposal pool leaving the gate fewer proxy-only edits to revert. The full combination achieves the highest ρ_T^{audit} at the lowest cluster decay, with in-loop gain within 2% of VANILLA AR.

Framework / Method	S_T	ρ_T^{audit}	H_T	Δ_T^{faith}
`autoresearch` *(Karpathy, 2026)*				
VANILLA AR	$0.66_{\pm.06}$	$0.46_{\pm.07}$	$1.18_{\pm.08}$	$0.171_{\pm.018}$
DAPS	$0.66_{\pm.05}$	$0.80_{\pm.05}$	$1.71_{\pm.07}$	$0.067_{\pm.010}$
Aider (Gauthier, 2024)				
VANILLA-AIDER	$0.74_{\pm.07}$	$0.48_{\pm.08}$	$1.22_{\pm.09}$	$0.165_{\pm.020}$
DAPS-AIDER	$0.75_{\pm.06}$	$0.76_{\pm.06}$	$1.65_{\pm.08}$	$0.072_{\pm.012}$

Table 6: Surface diversity S_T, faithfulness ratio ρ_T^{audit}, mechanism entropy H_T, and faithfulness gap Δ_T^{faith} at T=300 on T2 for the two ARL frameworks (`autoresearch` and Aider), each evaluated under a vanilla baseline and the DAPS variant. Subscripts denote ± 1 s.d. over 3 seeds.

4.6 Robustness Across Proposer LLMs

Figure 3 repeats T2 with three proposer LLMs. The pattern is uniform: every proposer exhibits algorithmic mode collapse under VANILLA AR, and DAPS reduces both mechanism-entropy collapse and the faithfulness gap for each. Llama-3.3-70B shows the most severe baseline collapse, consistent with its narrower prior over ML edits, while Claude Opus 4.7 produces the highest in-loop gain at every iteration count. Besides, the relative ranking of methods (Table 4) is preserved across proposers, suggesting that our findings are not artifacts of any single model's idiosyncrasies.

4.7 Robustness Across ARL Frameworks

Since `autoresearch` and Aider differ in how they generate, scope, and commit edits, a reasonable concern is that the collapse pattern we report is an artifact of `autoresearch`'s direct-diff proposer rather than a property of code-level ARL in general. To address this, we re-ran T2 with our Aider-based loop under both VANILLA and DAPS configurations, holding executor, acceptance rule, history summary, and diagnostic instrumentation fixed (§4.1.4). Results are provided in Table 6.

We have three observations as follows. *First*, under VANILLA-AIDER, the faithfulness ratio

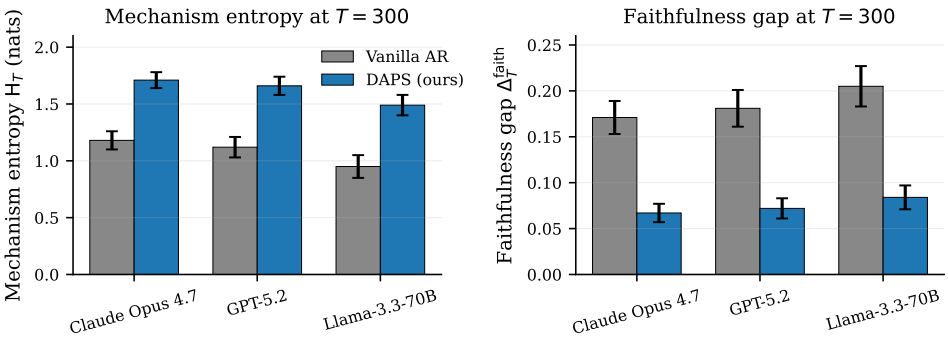

Figure 3: Mechanism entropy H_T (left) and faithfulness gap Δ_T^{faith} (right) at T=300 on T2, evaluated under VANILLA AR and DAPS for three proposer LLMs. Error bars show ± 1 s.d. over 3 seeds.

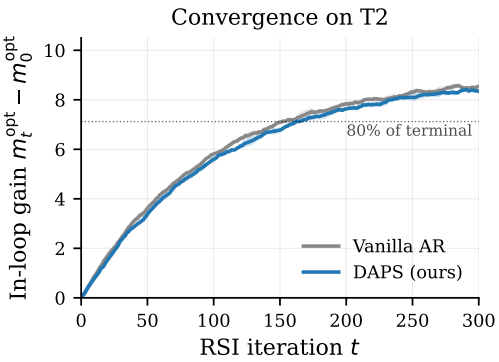

Figure 4: In-loop gain on T2 over T=300 iterations for VANILLA AR and DAPS. Mean$\pm$s.d. over 3 seeds; dotted line marks 80% of VANILLA AR's terminal gain.

($\rho_T^{\text{audit}} = 0.48$) and mechanism entropy ($H_T = 1.22$ nats) match VANILLA AR to within one standard deviation, despite Aider's edits being structurally different at the diff level. The surface diversity S_T is in fact higher under Aider (0.74 vs. 0.66) because its multi-line SEARCH/REPLACE blocks span more tokens, yet semantic and mechanism diversity track `autoresearch` closely. This is exactly the dissociation predicted by §4.4: surface differences between frameworks do not translate into differences in what the agent is actually *trying* to do. *Second*, DAPS transfers without modification: DAPS-AIDER improves ρ_T^{audit} from 0.48 to 0.76 and lifts mechanism entropy by 0.43 nats, mirroring the effect on `autoresearch`. *Third*, the small residual gap between DAPS-AIDER and DAPS (ρ_T^{audit} 0.76 vs. 0.80) is consistent with Aider's coarser-grained edits being slightly harder to deduplicate at the description level. Overall, algorithmic mode collapse and its mitigation are not framework-specific.

4.8 Efficiency Analysis

Here, we demonstrate that DAPS also matches VANILLA AR's convergence rate, and that its three components add bounded wall-clock cost. Figure 4 plots m_t^{opt} over T2's full trajectory: the two methods are statistically indistinguishable through $\sim$150 iterations and remain within one standard deviation at T=300. The number of iterations to reach 80% of VANILLA AR's terminal gain is $154_{\pm 8}$ for VANILLA AR and $163_{\pm 9}$ for DAPS, a difference close to the seed-to-seed variance. Convergence on T1, T3, and T4 follows the same pattern (Appendix B). On per-iteration wall-clock, CCR reuses each proposer-emitted edit description through a rule-based parser, PEM performs one Sentence-BERT embedding plus a 200-vector nearest-neighbor lookup, and HVG amortizes one audit evaluation across K=10 iterations. Aggregated against the 5-minute executor budget, DAPS adds approximately 1.4% to per-iteration wall-clock on T2, with the amortized HVG cost accounting for the bulk. Per-component breakdowns and cross-task timings are in Appendix B.

5 Conclusion and Future Work

In this paper, we illustrate that code-level autonomous research loops, despite using executable external metrics, exhibit algorithmic mode collapse: surface edit diversity remains intact while semantic and mechanism-level diversity progressively concentrate, and in-loop gains decreasingly transfer to held-out evaluation. A four-axis diagnostic instrument makes the phenomenon measurable, and a lightweight intervention DAPS substantially mitigates it without harming optimization speed. Under a three-tier protocol separating the in-loop metric, the audit signal read by the gate, and a blind evaluation no component accesses, the advantage persists on data the loop never influenced. Future work could examine collapse dynamics under frontier-scale executors and much longer horizons, extend the diagnosis beyond the preliminary non-ML loop, and study how multi-agent loops alter the picture.

Limitations

Our study is confined to model and data scales accessible on a single A100 machine; whether algorithmic mode collapse looks the same when proposer and executor models are both at the frontier, or when budgets allow 10k+ iterations, remains open, although the 1,000-iteration run of Appendix L shows the collapse deepening rather than self-correcting. Our headline claims are scoped to code-level ARLs that edit ML pipelines: the performance-engineering loop of Appendix M is a single preliminary probe outside that scope, and our mechanism taxonomy is ML-centric and would need re-derivation to study ARL in non-ML domains (e.g., theorem proving, scientific simulation), even though the derivation protocol of §3.2 is domain-general. The audit metric read by HVG is a validation signal rather than an untouched final test, which is why we report blind evaluations as headline numbers; those evaluations, though independent, are themselves benchmarks and may share idiosyncratic biases with the in-loop metric. Finally, corpus similarity V_T is descriptive and agrees only moderately with human novelty judgements (Appendix K), so it should not be read as direct evidence of retrieval from pretraining.

Ethical Considerations

Autonomous research agents that appear to improve themselves but in fact overfit their own evaluations pose a misinformation risk in scientific contexts. Our diagnostic instrument and DAPS are intended as cautionary tools: they should not be read as endorsements of unrestricted ARL deployment, but as steps toward making such systems more transparent about whether their claimed gains generalize. All artifacts we plan to release are training-pipeline edits and analysis scripts; no personal or sensitive data is involved. We use only public benchmarks under their respective licenses. The human annotation studies reported in Appendices I and K were conducted by volunteer researchers who consented to the use of their labels and who annotated only code-edit descriptions containing no personal data.

References

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. Databricks Blog.

John M Ford. 2004. Content analysis: An introduction to its methodology. *Personnel psychology*, 57(4):1110.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science*, 315(5814):972–976.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pages 10835–10866. PMLR.

Paul Gauthier. 2024. Aider: Ai pair programming in your terminal. GitHub repository. https://github.com/Aider-AI/aider.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, and 1 others. 2025. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tarun Gupta and Danish Pruthi. 2025. All that glitters is not novel: Plagiarism in ai generated research. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25721–25738.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Andrej Karpathy. 2026. autoresearch: Ai agents running research on single-gpu nanochat training automatically. GitHub repository. https://github.com/karpathy/autoresearch.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, and 1 others. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Grgur Kovač, Jérémy Perez, Rémy Portelas, Peter Ford Dominey, and Pierre-Yves Oudeyer. 2025. Recursive training loops in llms: How training data properties modulate distribution shift in generated data? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32278–32297.

Lyudmyla F Kozachenko and Nikolai N Leonenko. 1987. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Gengsheng Li, Jinghan He, Shijie Wang, Dan Zhang, Ruiqi Liu, Renrui Zhang, Zijun Yao, Junfeng Fang, Haiyun Guo, and Jinqiao Wang. 2026. R-diverse: Mitigating diversity illusion in self-play llm training. *arXiv preprint arXiv:2602.13103*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models.

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

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Vaibhav Mishra. 2026. Preventing curriculum collapse in self-evolving reasoning systems. *arXiv preprint arXiv:2603.13309*.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. 2022. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1525–1534.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward gaming. *Advances in neural information processing systems*, 35:9460–9471.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2026. Ai-researcher: Autonomous scientific innovation. *Advances in Neural Information Processing Systems*, 38:9481–9520.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Dhruv Trehan and Paras Chopra. 2026. Why llms aren't scientists yet: Lessons from four autonomous research attempts. *arXiv preprint arXiv:2601.03315*.

Ry Walker. 2026. Autoresearch tools. Online. https://rywalker.com/research/autoresearch-tools.

Lei Xiong, Kun Luo, Ziyi Xia, Wenbo Zhang, Jin-Ge Yao, Zheng Liu, Jingying Shao, Jianlyu Chen, Hongjin Qian, Xi Yang, and 1 others. 2026. Autoresearchbench: Benchmarking ai agents on complex scientific literature discovery. *arXiv preprint arXiv:2604.25256*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Bingze Zhu and Yubo Xie. 2026. Countering model collapse in iterative self-training via dynamic center-edge sampling. *Electronics*, 15(4):869.

A A Conceptual Model of Algorithmic Mode Collapse

We sketch a conceptual account that predicts when and why a code-level autonomous research loop should undergo algorithmic mode collapse, framing the four diagnostic axes (§3.2) and the DAPS components (§3.4) as instruments for testing its predictions.

Let $\pi_\theta(\rho, c \mid \mathcal{P}, h)$ denote the proposer's joint distribution over patches ρ and their mechanism categories c, conditioned on the current pipeline $\mathcal{P}$ and history h. Three structural facts about this distribution drive the dynamics.

(F1) Non-uniform prior over mechanism categories. Pretraining and instruction data over-represent some ML edit types: learning-rate adjustment, optimizer hyperparameter changes, simple regularization tweaks. Meanwhile, they under-represent others, such as custom kernels, packed-sequence handling, or idiosyncratic loss formulations. Marginalizing π_θ over patches yields a heavily peaked categorical $\pi_\theta(c \mid \mathcal{P})$.

(F2) Category-dependent acceptance rate. The acceptance rule R retains an edit only when its measured gain exceeds a noise floor ϵ. Different categories have systematically different probabilities of producing edits with reliable signal above ϵ: small-step micro-tunings (scheduling, regularization) move m^{opt} by small but consistent amounts, while structural edits (architecture, loss reformulation) move it more in expectation but with high variance and frequent regressions. Writing $\alpha(c)$ for the per-category acceptance rate, the empirical distribution of *accepted* edits is proportional to $\pi_\theta(c) \cdot \alpha(c)$, already a sharpening of $\pi_\theta(c)$ toward low-variance categories.

(F3) Self-imitation through history conditioning. Code-level autonomous research loops typically include accepted edits in h. The conditional $\pi_\theta(c \mid \mathcal{P}, h_t)$ then drifts toward the empirical distribution of past acceptances, a positive-feedback loop that further concentrates probability mass on the $\pi_\theta(c) \cdot \alpha(c)$-favoured corner.

Consequence. Iterating (F1) to (F3), the system converges on a narrow subset of mechanism categories, those jointly favoured by the proposer's prior and the acceptance noise floor, *independently of whether those categories contain the largest true improvements*. Inside the loop this manifests as continued growth of m^{opt} through increasingly fine-grained edits within a shrinking category set. Outside the loop, the audit and blind metrics improve only insofar as the favoured categories happen to contain genuine capability-affecting edits, a property the dynamics does not select for. The faithfulness gap Δ_t^{faith} (Eq. 1) therefore widens monotoni-

cally.

Testable predictions. The account yields six predictions that organize the empirical study:

P1. Surface diversity S_t is largely orthogonal to collapse, since $\pi_\theta(c)$ can sharpen without lexical repetition of diffs.

P2. Proposers whose pretraining yields broader categorical priors should collapse later and less deeply.

P3. Raising proposer sampling temperature redistributes mass within categories but does not alter $\pi_\theta(c)$, and should therefore not arrest collapse.

P4. Memory-based deduplication of recently accepted edits prevents semantic-cluster cycling but leaves $\pi_\theta(c)$ untouched; it should reduce ΔC only modestly.

P5. Interventions that directly re-weight the effective $\pi_\theta(c)$, category-coverage incentives, attack (F1) at its root and should yield larger reductions in ΔC.

P6. Held-out audits are the only mechanism that can undo edits that exploit the gap between the in-loop and the audited metric, since by construction the gap is invisible inside the loop.

The four diagnostic axes operationalize the antecedent of each prediction: S_t for P1, H_t together with C_t and V_t for P2 to P5, and Δ_t^{faith} for P6. The three DAPS components correspond directly: PEM addresses (P4), CCR addresses (P5), and HVG addresses (P6). The absence of an analogous DAPS component for (P3) is itself a prediction of the model. That is, raising temperature is not a designed intervention because the account claims it would not help. The longer-horizon run of Appendix L is a further test: because (F1) to (F3) compound with iteration count, the account predicts deeper rather than self-correcting collapse, which is what we observe.

B Efficiency: Per-Component Overhead and Cross-Task Convergence

Table 7 reports the per-iteration timing decomposition on T2. HVG dominates the DAPS overhead at $\sim$3.5 seconds per iteration (amortized 35-second audit evaluations every K=10 iterations); CCR and PEM are negligible. The single blind evaluation per run adds 35 seconds once, i.e. 0.02% of a 300-iteration budget. On the other three tasks, the total overhead is similar in absolute terms but varies modestly as a percentage with audit evaluation length: 1.4% on T1 (validation perplexity over a 0.5B-token shard takes longest), 1.2% on T3, and 0.7% on T4.

Component	ms / iter	% wall-clock
Executor (5-min budget)	300,000	97.3
Proposer LLM call	4,100	1.33
CCR (classify N=8 cands.)	410	0.13
PEM (embed + NN over M=200)	260	0.08
HVG (amortized every K=10)	3,500	1.14
DAPS overhead (sum)	4,170	1.35

Table 7: Per-iteration wall-clock decomposition on T2 with Claude Opus 4.7 as proposer and a single A100 executor. Times averaged over a 300-iteration run.

Convergence trajectories on T1, T3, and T4 mirror Figure 4: DAPS and VANILLA AR are statistically indistinguishable through the first $\sim$150 iterations and remain within one standard deviation of each other at T=300. Per-task scalar summaries (iterations to reach 80% of VANILLA AR's terminal gain) are $187_{\pm 14}$ vs $192_{\pm 16}$ on T1, $108_{\pm 9}$ vs $109_{\pm 10}$ on T3, and $96_{\pm 7}$ vs $98_{\pm 8}$ on T4.

C Hyperparameter Sensitivity

Table 8 reports the sensitivity of DAPS to each of its hyperparameters, with the remaining values fixed at the main-experiment defaults and the proposer fixed to Claude Opus 4.7. All rows are averaged over 3 seeds on T2; per-row standard deviations are below 0.05 on ρ_T^{audit} and below 1.5 on $\bar{g}_T^{\text{opt}}$. Defaults are highlighted in bold.

Five qualitative observations on Table 8 guide hyperparameter selection in practice. *First*, τ_m exhibits a sharp failure mode at 0.95: PEM rejects almost no candidates and ρ_T^{audit} collapses back toward the VANILLA AR value (0.46), confirming that semantic deduplication carries most of the diversity recovery rather than the other two components alone. *Second*, τ_m=0.75 damages in-loop gain by starving the proposer (too many candidates rejected before reaching the executor) without commensurately improving faithfulness. *Third*, M is comparatively forgiving in the central range, but M=50 allows semantic cycling to re-emerge within a W=20 window and M=500 begins to over-suppress legitimate revisits of previously useful edit categories. *Fourth*, τ_c is essentially flat in the range tested, since the category-coverage reweighting acts as a soft prior on rare categories whose exact temperature matters little. *Fifth*, τ_h

Value	ρ_T^{audit}	$\bar{g}_T^{\text{opt}}$	ΔC
PEM *similarity threshold* τ_m			
0.75	0.71	+7.7	0.18
0.80	0.78	+8.6	0.22
0.85	**0.80**	**+8.8**	**0.21**
0.90	0.77	+8.9	0.26
0.95	0.51	+8.9	0.55
PEM *memory size* M			
50	0.68	+8.7	0.34
100	0.77	+8.8	0.25
200	**0.80**	**+8.8**	**0.21**
300	0.79	+8.8	0.20
500	0.74	+8.6	0.19
CCR *temperature* τ_c			
0.5	0.79	+8.8	0.23
1.0	**0.80**	**+8.8**	**0.21**
1.5	0.79	+8.7	0.22
HVG *percentile* τ_h			
70th	0.83	+8.6	0.20
80th	**0.80**	**+8.8**	**0.21**
90th	0.78	+8.9	0.24
HVG *audit interval* K			
5	0.81	+8.7	0.21
10	**0.80**	**+8.8**	**0.21**
20	0.77	+8.8	0.24

Table 8: Hyperparameter sensitivity of DAPS on T2. Defaults used in the main experiments are bold.

and K trade in-loop gain against faithfulness in opposite directions: aggressive auditing (τ_h=70th, K=5) slightly raises ρ_T^{audit} at the cost of $\bar{g}_T^{\text{opt}}$, while loose auditing (τ_h=90th, K=20) preserves in-loop gain but lets Goodhart edits accumulate between audits. The chosen defaults sit near the knee of both trade-offs.

D Qualitative Analysis of Collapse

Inspecting the accepted-edit logs reveals a recurring pattern under VANILLA AR. On T1, the first $\sim$50 iterations show a broad mix of optimizer, architectural, and data-mixing edits. By iteration 200, 74% of accepted edits modify the learning-rate schedule or AdamW hyperparameters; many are micro-tunings (e.g., warmup steps 200 $\rightarrow$ 220, β_2 0.95 $\rightarrow$ 0.96) that nudge the in-loop validation loss without affecting LAMBADA or C4 perplexity. On T3, a striking 61% of late-stage edits are variants of "increase chain-of-thought temperature" or "add an arithmetic-only loss term"; audited ARC-Easy accuracy is flat under these. Under DAPS, the edit stream remains qualitatively heterogeneous through iteration 300: representative late-stage edits on T1 include "swap RMSNorm for LayerNorm with re-tuned ϵ," "insert a curriculum filter on token-length," and "modify attention-mask handling for packed sequences." These are not necessarily better edits, but they sample the edit space rather than concentrating on a single mode.

E Qualitative Edit Examples

To complement the quantitative collapse signals reported in §4.4, we present representative accepted-edit diffs sampled from the T1 logs (single-GPU `nanochat`-style pretraining on the `autoresearch` framework of Karpathy (2026), where the agent is permitted to modify only `train.py`). Hunk headers report iteration index and the mechanism category assigned by the procedure of §3.2. Diffs are paraphrased and trimmed to a few lines of relevant context for legibility. The full version of verbatim logs will be released.

E.1 Early-Phase Diversity Under VANILLA AR

In the first $\sim$60 iterations the proposer explores a broad set of mechanism categories. The four examples below come from a single seed of VANILLA AR on T1 and span four distinct categories of the taxonomy in Table 1.

```
@@ train.py: iteration 14 (optimizer) @@
     optimizer = torch.optim.AdamW(
         model.parameters(),
-        lr=3e-4, betas=(0.9, 0.95),
+        lr=3e-4, betas=(0.9, 0.97),
         weight_decay=0.1, fused=True,
     )
```

```
@@ train.py: iteration 27 (architecture) @@
 class Block(nn.Module):
     def __init__(self, config):
         super().__init__()
-        self.ln_1 = nn.LayerNorm(config.n_embd)
-        self.ln_2 = nn.LayerNorm(config.n_embd)
+        self.ln_1 = RMSNorm(config.n_embd, eps=1e-6)
+        self.ln_2 = RMSNorm(config.n_embd, eps=1e-6)
```

```
@@ train.py: iteration 39 (data) @@
-block_size  = 1024
-batch_size  = 32
+block_size  = 2048     # longer context, fewer sequences
+batch_size  = 16
```

```
@@ train.py: iteration 51 (regularization) @@
         self.c_proj = nn.Linear(4 * n_embd, n_embd,
      bias=False)
-        self.dropout = nn.Dropout(0.0)
+        self.dropout = nn.Dropout(0.05)
```

E.2 Late-Phase Collapse Under VANILLA AR

The five edits below come from the same trajectory between iterations 250 and 290. All five fall into the *optimizer* or *scheduling* category despite editing different identifiers on different lines of `train.py`. This is the qualitative signature underlying the entropy collapse H_{300}=1.18 nats reported in Table 4:

surface diversity is preserved while mechanism diversity is not.

```
@@ train.py: iteration 257 (scheduling) @@
-warmup_iters   = 200
+warmup_iters   = 240
 lr_decay_iters = 900
 min_lr         = 3e-5
```

```
@@ train.py: iteration 263 (optimizer) @@
-        lr=3e-4, betas=(0.9, 0.97),
+        lr=3e-4, betas=(0.9, 0.96),
         weight_decay=0.1, fused=True,
```

```
@@ train.py: iteration 271 (scheduling) @@
 warmup_iters   = 240
-lr_decay_iters = 900
-min_lr         = 3e-5
+lr_decay_iters = 950
+min_lr         = 1e-5
```

```
@@ train.py: iteration 283 (optimizer) @@
     optimizer = torch.optim.AdamW(
         model.parameters(),
-        lr=3e-4,  betas=(0.9, 0.96),
-        weight_decay=0.10, fused=True,
+        lr=2.8e-4, betas=(0.9, 0.96),
+        weight_decay=0.08, fused=True,
     )
```

```
@@ train.py: iteration 289 (scheduling) @@
-warmup_iters   = 240
+warmup_iters   = 220
 lr_decay_iters = 950
```

E.3 Late-Phase Diversity Under DAPS

The five edits below come from a DAPS trajectory on the same task, sampled in the same iteration window (250 to 290). They span five distinct mechanism categories. CCR and PEM do not prohibit any category; they re-weight rare categories upward and suppress near-duplicates of recently accepted edits. Scheduling edits still appear (e.g., iteration 290) but no longer dominate the window.

```
@@ train.py: iteration 254 (data) @@
 def stream_batches(tokens, block_size):
+    # drop pathologically short sequences from the stream
+    if len(tokens) < 96:
+        return
     yield pack_into_blocks(tokens, block_size)
```

```
@@ train.py: iteration 266 (architecture) @@
 class MLP(nn.Module):
     def forward(self, x):
-        return self.c_proj(F.gelu(self.c_fc(x)))
+        a = self.c_fc(x)
+        b = self.c_gate(x)              # SwiGLU branch
+        return self.c_proj(F.silu(a) * b)
```

```
@@ train.py: iteration 275 (numerical) @@
         att = (q @ k.transpose(-2, -1)) * scale
-        att = F.softmax(att, dim=-1)
+        # softmax in fp32 for numerical stability
+        att = F.softmax(att, dim=-1,
      dtype=torch.float32).to(q.dtype)
```

```
@@ train.py: iteration 283 (regularization) @@
         self.c_proj = nn.Linear(4 * n_embd, n_embd,
      bias=False)
-        self.dropout = nn.Dropout(0.0)
+        self.dropout = nn.Dropout(0.05)
```

```
@@ train.py: iteration 290 (scheduling) @@
-warmup_iters   = 200
+warmup_iters   = 250
 lr_decay_iters = 900
```

E.4 Connection to the Diagnostic Axes

The contrast between Appendices E.2 and E.3 illustrates each axis introduced in §3.2: surface diversity S_t is similar in both panels (lines edited, tokens touched, and structural shape of the hunks are comparable); semantic cluster count C_t and mechanism entropy H_t are clearly lower under VANILLA AR; and corpus similarity V_t is visibly higher under VANILLA AR since LR-schedule micro-tunings closely match the most common edit pattern in publicly available ML training repositories. We selected the windows above to be representative rather than extremal. The complete logs containing the full trajectories from which these hunks were sampled will also be released.

F Summary of Validity, Robustness, and Scope Checks

Table 9 indexes the checks on which the claims of §4.2 and §4.3 rest. Each row names a threat to those claims, the check that addresses it, and the appendix that reports the outcome; the numbers appear only in the referenced appendix and are not restated here. Across every check the direction of the collapse and the ordering VANILLA AR $<$ diversity baselines $<$ DAPS are preserved. On this basis we scope the headline claims to ARLs that edit ML pipelines at single-GPU executor scale over horizons up to 1,000 iterations, and treat evidence outside that setting as preliminary.

G Three-Tier Evaluation Protocol: Roles, Per-Task Results, and Absolute Values

This appendix specifies the information flow among the three evaluations of §3.3 and reports the results that Table 3 summarizes.

Roles and partitions. Table 10 lists the exact dataset or partition used for each role and task. Only m^{opt} is visible to the proposer. m^{audit} is read by HVG once every $K{=}10$ iterations and by the calibration of τ_h over the first 30 iterations, in both cases through a scalar comparison; the proposer never observes audit values, per-example outcomes, or the identity of audit examples, and the failure notice appended to h is templated and value-free. m^{blind} is evaluated once, after the run terminates,

Threat to the claims, and the check addressing it	App.
Does ρ track Goodharting rather than benchmark mismatch? Known-good and proxy-only control edits applied in isolation	H
Could benign distribution shift alone explain $\rho_T < 1$? Non-adaptive RANDSEARCH on the same task pairs, and the shape of Δ_t^{faith}	H
Does the gate leak audit information into the loop? Blind sets read by no component at any point	G
Is the collapse an artifact of the summarizer, the embedding model, or the clustering? Systematic pipeline variations, plus human labels	I
Is it an artifact of the mechanism taxonomy? Coarser, finer, random, and human taxonomies, plus a taxonomy-free entropy estimate	J
Does any conclusion require an LLM in the measurement path? An AST-based detector and the metric-only Δ_t^{faith}	I
Is corpus similarity a meaningful axis? Human novelty ratings against V_t, and summarizer sensitivity	K
Does the collapse self-correct over longer horizons, or vanish at larger target scale? T=1000 and 8B-target runs	L
Does the phenomenon appear outside ML pipelines? A preliminary runtime-optimization loop on a log-analytics pipeline	M
Are the baselines tuned comparably to DAPS? Search grids and the shared tuning budget	N

Table 9: Index of the validity, robustness, and scope checks. The right column gives the appendix in which each result is reported.

by a separate offline script that has no channel back into the loop. Baselines B1 to B6 consume no audit signal, so for them both ρ_T^{audit} and ρ_T^{blind} are pure post-hoc evaluations; B7, B8, and DAPS consume the audit signal at the same cadence, cost, and feedback format.

Per-task faithfulness. Table 11 gives ρ_T^{audit} and ρ_T^{blind} per task at T=300 over 3 seeds. DAPS retains its advantage on sets it has never influenced, and VANILLA AR, which consumes no audit signal, shows an audit-to-blind drop of comparable size, so the drop is attributable to benchmark idiosyncrasy rather than to leakage through the gate. The gate fires 2.7 ± 0.9 times per 300-iteration run.

Absolute values. Table 12 reports absolute audit-metric values so that the practical magnitude of the movements can be judged directly. For T1 these correspond to log-perplexity reductions of 0.058 nats (VANILLA AR) and 0.108 nats (DAPS) against in-loop loss reductions of 0.142 and 0.139, reproducing the T1 ratios of Table 2. In-loop absolutes follow the same pattern: T2 dev win-rate rises from 38.4 to 47.3 (VANILLA AR) and 47.2 (DAPS), and T3 GSM8K dev exact match from 31.2 to 42.9 and 42.7.

	m^{opt} (in-loop)	m^{audit} (gate only)	m^{blind} (never read)
T1	OpenWebText held-in validation loss	LAMBADA and held-out C4 perplexity	WikiText-103 log-perplexity
T2	Win-rate on 500 Alpaca dev prompts	MMLU 5-shot and IFEval	Win-rate on 500 Dolly instructions
T3	GSM8K dev exact match	ARC-Easy and MATH-500	SVAMP accuracy
T4	ARC-Easy dev split accuracy	ARC-Challenge and CommonsenseQA	OpenBookQA accuracy

Table 10: Datasets and partitions used for the three evaluation roles. All three are disjoint within a task.

Task	VANILLA AR		DAPS	
	ρ_T^{audit}	ρ_T^{blind}	ρ_T^{audit}	ρ_T^{blind}
T1	0.41	0.39	0.78	0.73
T2	0.46	0.42	0.80	0.76
T3	0.38	0.35	0.74	0.69
T4	0.49	0.46	0.82	0.78
Average	0.44	0.41	**0.79**	**0.74**

Table 11: Audited and blind faithfulness ratios at T=300, mean over 3 seeds. The audit columns reproduce Table 2; the blind columns are new.

Audit set	t=0	Vanilla	DAPS
T1 LAMBADA perplexity	45.6	43.0	40.8
T1 C4 perplexity	28.8	27.2	25.9
T2 MMLU 5-shot (%)	31.4	35.0	38.1
T2 IFEval (%)	29.0	33.6	36.4
T3 ARC-Easy (%)	68.2	72.7	78.3
T3 MATH-500 (%)	10.4	14.8	17.2
T4 ARC-Challenge (%)	46.1	51.2	54.1
T4 CommonsenseQA (%)	58.7	63.0	65.9

Table 12: Absolute audit-metric values. The t=0 column is a single evaluation of the shared initial pipeline; the other two columns are means over 3 seeds at T=300.

H Construct-Validity Controls

To check ρ_T and Δ^{faith} measure Goodharting rather than benchmark mismatch, we ran two control sets whose ground truth is known by construction.

Positive controls. We curated 12 known-good edits from published recipes, including the nanoGPT speedrun lineage and open post-training changelogs: rotary position embeddings, a SwiGLU MLP, fused optimizer kernels that raise tokens processed per fixed budget, sequence-

length filtering, and corrected initialization scaling, among others. Each was applied in isolation to the T1 and T3 starting pipelines.

Negative controls. We constructed 8 deliberately proxy-only edits: dev-shard-specific example ordering on T1, judge-phrasing tweaks on T2, and dev-tuned answer-format templates on T4.

Control set	n	ρ (mean $\pm$ s.d.)
Known-good (published recipes)	12	0.93 ± 0.06 (min 0.82)
Proxy-only (dev-specific)	8	0.11 ± 0.08

Table 13: Per-edit faithfulness ratio for the two control sets, measured on the corresponding audit metrics.

Table 13 shows that under our task pairs ρ separates genuine improvement ($\rho \approx 1$) from proxy-only improvement ($\rho \approx 0$), which is exactly the construct the main claims require. Two further observations argue against benign distribution shift as an explanation of $\rho_T < 1$ inside the loop. First, RANDSEARCH shares the identical task pairs but selects edits non-adaptively and attains 0.74 to 0.81 (Table 2), so the pairs themselves support near-full transfer and the low ρ_T under VANILLA AR is induced by adaptive optimization. Second, a static mismatch would produce a constant offset, whereas Δ_t^{faith} widens monotonically within a fixed task pair (Figure 2). Finally, each task carries two independent audit components; reporting ρ per component, the ordering VANILLA AR $<$ diversity baselines $<$ DAPS holds on all 8 components individually (VANILLA AR range 0.33 to 0.52; DAPS 0.69 to 0.85) and on the 4 blind sets of Appendix G, i.e. on 12 independent evaluations in total.

I Robustness of the Diagnostic Pipeline

The collapse signal should not hinge on any single measurement choice. We therefore recomputed cluster decay ΔC on T2 under systematic variations of the summarizer, the embedding model, and the clustering algorithm (Table 14). Per-window C_t trajectories correlate at Spearman > 0.93 across all variants, and the VANILLA AR against DAPS gap is preserved in every row.

Human labels. Three annotators labelled 300 sampled edits with the categories of Table 1 (majority vote; $\kappa = 0.81$ against our rule-based parser, in line with the parser-against-LLM $\kappa = 0.84$ of §3.2). Mechanism entropy computed from the human labels drops from 1.72 to 1.21 nats under VANILLA AR, closely tracking the parser-based 1.76 to 1.18 of Figure 2.

Variation of the diagnostic pipeline	ΔC
Main config: fixed summarizer, `all-mpnet-base-v2`, HDBSCAN (mcs=5)	0.68 (0.21)
Summarizer swapped to GPT-5.2	0.66 (0.22)
Summarizer swapped to Llama-3.3-70B	0.70 (0.24)
Embedding: `all-MiniLM-L6-v2`	0.65 (0.20)
Embedding: `e5-large-v2`	0.69 (0.21)
HDBSCAN mcs=3 / mcs=8	0.71 / 0.63 (0.23 / 0.19)
k-means, k chosen per window by silhouette	0.62 (0.18)

Table 14: Cluster decay on T2 for VANILLA AR, with DAPS in parentheses.

A measurement with no LLM in the path. An AST-based rule detector applied directly to the raw code diffs reproduces the Appendix D statistic: 71% of late-stage T1 edits touch optimizer or learning-rate-schedule code, against 74% for the parser-based count. Independently, Δ_t^{faith} uses only executed metric values, with no summarizer, embedding, clustering, or taxonomy anywhere in its computation, and it widens exactly when C_t and H_t collapse (Figure 2, Table 4).

J Taxonomy Variants and a Taxonomy-Free Estimate

A hand-designed taxonomy could in principle place category boundaries so as to manufacture the entropy drop, even though ours was frozen before our runs (§3.2). We therefore recomputed H_t on T2 under four alternative taxonomies. *Coarse*: four super-categories merging Table 1 (Optimization = Optimizer + Scheduling; Model = Architecture + Numerical + Decoding; Data and Objective = Data + Loss + Regularization; Other). *Fine*: 17 subcategories obtained by re-clustering the pilot with a stricter merge criterion. *Random*: 10 taxonomies of nine categories each, obtained by randomly re-partitioning the 17 subcategories. *Human*: the 300 human-labelled edits of Appendix I.

Taxonomy variant	Vanilla drop	DAPS drop
Original (9 categories, Table 1)	0.58	0.08
Coarse (4 super-categories)	0.42	0.06
Fine (17 subcategories)	0.79	0.11
Random (10 draws, 9 categories)	0.51 ± 0.07	0.09 ± 0.03
Human labels (300 edits)	0.51	0.07

Table 15: Mechanism-entropy drop from t=30 to t=300 on T2, in nats.

Table 15 shows that the collapse and its mitigation are invariant to granularity and to ran-

dom boundary placement. Two taxonomy-free measurements corroborate this: the semantic cluster count C_t already reported in the main paper, and a Kozachenko-Leonenko k-nearest-neighbour differential-entropy estimate (Kozachenko and Leonenko, 1987) computed directly on the description embeddings, which drops by 0.44 nats under VANILLA AR and 0.07 nats under DAPS.

K Human Validation of Corpus Similarity

Corpus similarity V_t is a descriptive statistic (§3.2), and this appendix quantifies how well it tracks human judgement. We sampled 120 edit descriptions stratified by V_t quartile across tasks and asked three NLP researchers, blind to V_t and to the generating method, to rate each on a 1 to 5 scale for novelty relative to common ML practice (inter-rater Krippendorff $\alpha = 0.61$; Ford, 2004). The Spearman correlation between V_t and mean human novelty is -0.47 (bootstrap 95% CI -0.60 to -0.31), i.e. moderate validity in the expected direction. We additionally tested wording sensitivity by re-summarizing every description with a different LLM, which changes V_t by only 0.021 on average, so the statistic is not dominated by one summarizer's phrasing. Together these results support keeping V_t as a secondary axis reported with the caveats stated in §3.2, and not as evidence of retrieval.

L Horizon and Target-Scale Extensions

Table 16 extends T2 along the two axes flagged in the Limitations section, with one seed per new configuration. Over the longest horizon our budget allows, collapse deepens rather than self-corrects, which is what the conceptual model of Appendix A predicts, since the feedback mechanisms (F1) to (F3) compound with iteration count; DAPS remains stable over the same horizon. A first step up in target-model scale reproduces the 1B pattern. Note that the main study already includes frontier-scale proposers (§4.6), so the axes that remain open are executor and target scale, and horizons beyond 1,000 iterations.

M A Preliminary Non-ML Autonomous Research Loop

Our headline claims concern ARLs that edit ML pipelines. As a first probe outside that scope, we built a performance-engineering loop in which the agent edits a Python log-analytics pipeline to minimize wall-clock runtime, subject to exact-output correctness checks. Here m^{opt} is runtime on 20 development inputs, and m^{audit} is runtime on 40 inputs drawn from a different data snapshot together with the correctness suite. A new seven-category taxonomy (algorithmic complexity, data structures, batching and I/O, caching, parallelism, memory layout, micro-optimization) was derived with the same pilot protocol as §3.2, from public performance-tuning commits rather than ML logs.

With 150 iterations, 2 seeds, and the same proposer, we observe the same signature: surface diversity is flat (0.71 to 0.70) while semantic clusters fall from 11 to 5 ($\Delta C = 0.55$), 66% of late-stage accepted edits are caching or micro-optimizations, and ρ_T is 0.58 under VANILLA AR against 0.81 under DAPS. This suggests that neither the phenomenon nor the mitigation is specific to ML-pipeline editing, but the study is small and single-domain, so we report it as preliminary and keep the headline claims scoped as stated in §1.

Setting (T2)	ρ_T^{audit}		H_T	
	Vanilla	DAPS	Vanilla	DAPS
T=300, 1B target (main study)	0.46	0.80	1.18	1.71
T=1000, 1B target (new)	0.31	0.74	0.94	1.63
T=300, 8B target (new)	0.44	0.77	1.16	1.68

Table 16: Horizon and target-scale extensions on T2. The first row is reproduced from Tables 2 and 6; the other rows use one seed each.

N Baseline Tuning Grids

Every baseline received the same tuning budget as DAPS's own calibration: a three-point grid per key hyperparameter, evaluated on the first 50 iterations of T2, with the best setting then fixed for all tasks and seeds. The grids are: HITEMP proposer temperature $\{1.0, 1.2, 1.5\}$; R-DIVERSE-A similarity threshold $\{0.80, 0.85, 0.90\}$ and memory size $\{100, 200, 300\}$; PRISM-A cluster-coverage reward weight $\{0.1, 0.5, 1.0\}$; REFLEXION reflective-summary length $\{64, 128, 256\}$ tokens; HO-EARLYSTOP and HO-REVERT audit interval $K \in \{5, 10, 20\}$. RANDSEARCH has no tunable hyperparameter beyond the size of its harvested edit corpus, which we fixed at 200 modifications. For DAPS the corresponding sweep is reported in Appendix C.