# SEED: SEPARATING ADMISSION FROM DEPLOYMENT EVIDENCE IN ADAPTIVE CODING-AGENT UPDATES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Adaptive coding agents select prompt and workflow updates using evaluator feedback. Once a hidden split helps choose the surviving update, however, it no longer supplies independent deployment evidence. We introduce SEED, a four-stage protocol that separates visible ranking, hidden admission, disjoint canary confirmation, and deployment or rollback. Partitions are fixed within each run, not resampled across generations. The analysis spans 24 model lines, six benchmark families, two serving stacks, and all 432 model–benchmark–profile cells, while distinguishing retry-inclusive gate exposures from distinct incumbent–candidate decisions. Canary checks reject 25.1%, 25.3%, and 33.6% of first admissions in the Anchor, Horizon, and Confirmatory profiles. A seemingly large retry-inclusive Anchor–Horizon increase of $+12.26$ percentage points disappears after keeping only the first exact pair per run ($-0.73$ points, 95% interval $[-2.37, 0.91]$), so the raw contrast measures operational gate burden rather than increased distinct-decision non-transfer. Among 652 first-exact rollback decisions, 271 prevent a local untouched-final pass-or-gap regression and 159 prevent direct pass loss, but recall is about 33% and fixed-budget endpoint intervals all cross zero. We provide a bounded, retry-aware deployment-trace protocol and an empirical account of its behavior. The results do not show that a canary is an oracle, identify an optimal budget split, or establish contamination-free or non-coding transfer.

## 1 INTRODUCTION

Coding agents revise prompts, add self-review, change repair routines, or recombine workflow modules using evaluator feedback (Pryzant et al., 2023; Yang et al., 2024; Khattab et al., 2024; Opsahl-Ong et al., 2024; Agrawal et al., 2026; Robeyns et al., 2025; Zhang et al., 2026; Liu et al., 2025). Useful for search, this feedback creates a familiar statistical problem: an evaluator that shapes the chosen system is no longer untouched evidence about it. Reusable-holdout and leaderboard work formalize this danger under adaptive querying (Dwork et al., 2015b;a; Blum & Hardt, 2015). Coding agents make it operational: an update can clear hidden admission, become the incumbent, and still fail disjoint tasks evaluated only after admission.

SEED (Self-Evolution via Evaluation-as-Deployment) implements a four-stage protocol: visible tasks rank bounded prompt/workflow mutations, a hidden holdout provisionally admits an update, a disjoint canary checks it, and the system deploys, rolls back to the incumbent, or records a no-op. A final split is reserved for retrospective reporting. The stages are an offline analogue of production canarying and rollback (Warner & Davidovič, 2018), alongside established MLOps evaluation practices (Sculley et al., 2015; Breck et al., 2017; Zinkevich, 2017). The contribution is a trace schema and empirical study of history-conditioned coding-agent updates, not the invention of canarying or rollback.

The *event/decision* distinction is central. Rollback changes the incumbent and later proposals, allowing the same exact incumbent–candidate pair to recur, especially in longer search. Raw events measure gate workload; first-admission and first-exact-pair counts better characterize distinct decisions. Anchor and Horizon differ sharply at the retry-inclusive unit, but the first-exact-pair contrast is small and its interval crosses zero.

Our main findings are:



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

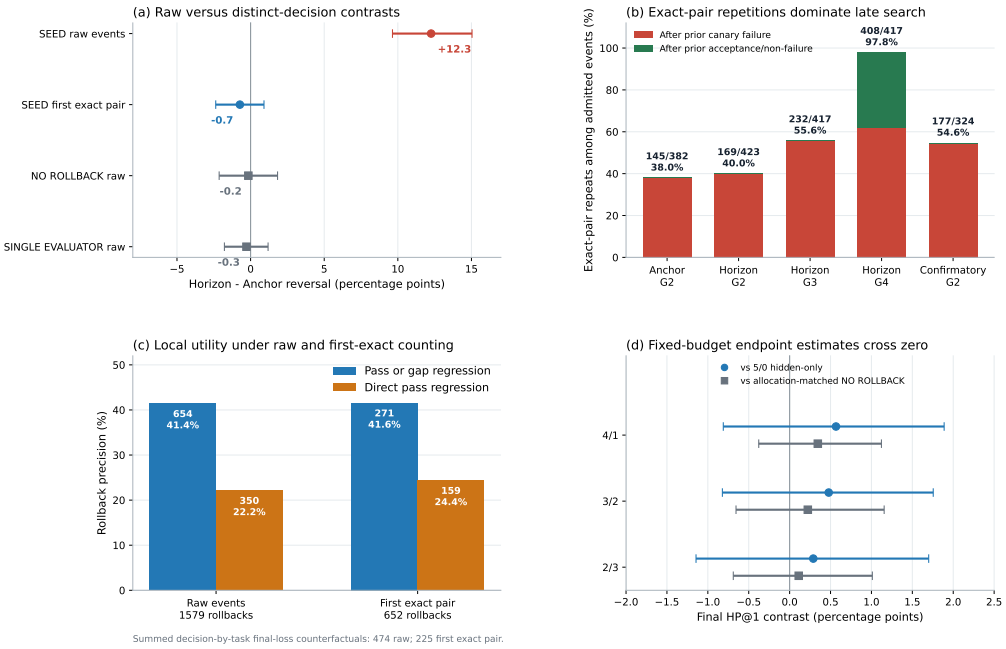


Figure 1: **Retry-aware evidence changes the interpretation.** (a) The positive retry-inclusive Anchor–Horizon contrast vanishes when each exact incumbent–candidate pair is counted once per run. (b) Exact-pair repeats dominate late Horizon search. (c) Rollback catches some local untouched-final regressions under either counting unit, but not all rollback events are protective. (d) Fixed-budget endpoint intervals cross zero. Intervals resample model clusters. Counts and scopes appear in Table 4 and Appendix C.

- **Canary confirmation often disagrees with provisional admission.** At first admission, disagreement is 145/577 (25.1%) for Anchor, 169/667 (25.3%) for Horizon, and 177/527 (33.6%) for Confirmatory. These are local, within-benchmark measurements, not contamination detection.
- **The raw Anchor–Horizon gap is retry-inclusive burden, not a distinct-decision effect.** Raw events give $+12.26$ percentage points (95% interval $[9.64, 15.04]$), but first exact pairs give $-0.73$ ($[-2.37, 0.91]$). Anchor and Horizon also change both search depth and proposal breadth, so neither contrast is a one-factor causal estimate.
- **Rollback has measurable but partial local utility.** Under first-exact counting, 271/652 rollbacks prevent a final pass-or-gap regression and 159/652 prevent direct final-pass loss. Corresponding recall is 33.9% and 33.3%; the gate is a partial detector rather than a certificate of downstream behavior.
- **Budget allocation is a trade-off rather than an endpoint optimum.** Across 57/60 self-hosted model–family lines, reallocating a fixed five-task budget changes admission, rollback, deployment, and compute, while every final-`HP@1` interval relative to hidden-only evaluation crosses zero.

SEED studies six-module, history-conditioned prompt/workflow search in coding, not open-ended self-modification. Canary task identities, examples, outcomes, and scores are excluded from mutation-proposal history, but rollback changes future state: the canary is *prompt-isolated at the proposal channel but system-adaptive*. Contamination, a genuinely matched split-size intervention, repository-scale rollback precision, and non-coding transfer remain unresolved.

## 2 Related Work

**Adaptive evaluation.** Optimizing a finite-sample selection criterion can bias subsequent performance estimates (Cawley & Talbot, 2010). Reusable holdouts and leaderboard mechanisms address

adaptive access to evaluation data (Dwork et al., 2015b;a; Blum & Hardt, 2015). Distribution shift and underspecification pose additional transfer problems (Recht et al., 2019; D'Amour et al., 2022; Koh et al., 2021). SEED does not inherit these mechanisms' statistical guarantees: it records which evidence selects a state and which evidence is reserved for reporting.

**Coding evaluation and freshness.** HumanEval, MBPP, their EvalPlus tests and Pro extensions, ClassEval, and APPS probe different functional behaviors (Chen et al., 2021; Austin et al., 2021; Liu et al., 2023; Yu et al., 2025; Du et al., 2023; Hendrycks et al., 2021). LiveCodeBench and SWE-bench extend coverage to time-indexed coding problems and repository issues (Jain et al., 2025; Jimenez et al., 2024). Stronger execution tests can reveal previously undetected errors (Liu et al., 2023); benchmark contamination is a separate concern (Magar & Schwartz, 2022; Deng et al., 2024). A disjoint canary does not establish training-data exclusion or temporal freshness.

**Agent optimization.** Prompt and LM-program optimizers select prompts, demonstrations, or programs using feedback (Pryzant et al., 2023; Yang et al., 2024; Khattab et al., 2024; Opsahl-Ong et al., 2024; Agrawal et al., 2026). Agent-design and self-improvement methods search broader workflow or code spaces (Hu et al., 2025; Robeyns et al., 2025; Zhang et al., 2026; Liu et al., 2025; Wang et al., 2026). VeRO provides versioned, budget-controlled evaluation and execution traces; HarnessOpt-Bench evaluates optimized harnesses on an inaccessible test partition (Ursekar et al., 2026; Anonymous, 2026). Our narrower question concerns the admission–confirmation boundary and the counting of repeated decisions within a bounded search.

**Deployment control and reporting.** Canarying and rollback predate agent optimization (Warner & Davidovič, 2018); evaluation-driven operations and governed upgrades likewise embed evaluation in a lifecycle (Xia et al., 2024; Qin et al., 2026). SEED neither invents these controls nor treats functional tests as a production safety certificate. Its trace complements model documentation and reproducibility (Mitchell et al., 2019; Pineau et al., 2021). Appendix M provides broader context, not evaluations of those systems.

# 3 Protocol

## 3.1 Four evidence roles

For model line $m$, benchmark family $b$, profile $p$, and seed $s$, four disjoint task roles separate evidence use: *tune* supports proposal ranking and bounded repair; *holdout* provisionally admits candidates; *canary*, evaluated only after admission, controls deployment; and *final* supports retrospective analysis only. The executor receives each task statement, but only tune identities, examples, outcomes, and scores enter mutation-proposal history. All variants within a cell share a manifest whose four partitions remain fixed across generations (Appendix A.1).

Table 1: **The four evidence roles.** Canary tasks are disjoint from tune and holdout, but fixed within-run reuse and rollback make the gate system-adaptive rather than independent across generations. The executor receives each evaluated task; the history column denotes feedback available to the mutation proposer.

| Stage | Data role | Enters proposal history? | Selects state? | Evidential status |
|---|---|---|---|---|
| Visible ranking | Tune | Yes | Proposes/ranks | Exploratory |
| Hidden admission | Holdout | No | Provisionally | Selection-exposed |
| Canary check | Canary | No | Deploy/rollback | Disjoint, adaptive |
| Final reporting | Final | No | No | Retrospective only |

Let $p_D(e)$ be pass rate for candidate $e$ on split $D$, and let $g_D(e) = p_{\text{visible}}(e) - p_D(e)$ be the visible–hidden gap. At the admission stage, candidates are ranked by

$$s(e;\theta) = 0.4p_{\text{visible}}(e) + 0.6p_{\text{hold}}(e) - 0.75[g_{\text{hold}}(e) - g_{\text{hold}}(\theta)]_+, \quad (1)$$

subject to hard admission gates. The weights are fixed globally and are not claimed optimal. The object of analysis is not the scalar score; it is the sequence of evidence roles and resulting deployment trace.

The mutation library contains six interpretable prompt/workflow choices: direct generation, edge-case planning, self-review, visible-test repair, a deliberately risky public-test-focused repair, and a conservative mixed policy. The library keeps trajectories inspectable but limits external validity. We compare Frozen, REFLECTION-ONLY, visible-only Unconstrained search, SINGLE EVALUATOR, NO ROLLBACK, SEED-FULL, and the more conservative SEED-TIGHT. Algorithm 1 gives the deployment procedure; Appendix A retains coefficient and variant details.

Table 2: The six prompt/workflow configurations. All return code only; the harness owns splitting, scoring, admission, canarying, and rollback.

| Library member | Mechanism | Role |
|---|---|---|
| `base_direct` | Direct code generation | Frozen incumbent/reference |
| `edge_case_planner` | Plans boundary cases before coding | Hidden-robustness scaffold |
| `self_review_guarded` | Draft, review, and rewrite | Reflection-style probe |
| `public_test_repair` | Bounded repair from visible failures | Transfer of visible repair |
| `balanced_seed` | Planning, review, cautious repair | Mixed candidate |
| `public_test_focused` | Targets visible failures aggressively | Deliberately risky stressor |

**Hard gates.** Admission requires non-decreasing visible tune and hidden-holdout pass and a non-widening hidden gap relative to incumbent $\theta$. The same three-part gate controls canary deployment versus rollback. SEED-TIGHT adds stricter margins, more often finding no admissible update. NO ROLLBACK computes canary outcomes but deploys regardless; this compares deployment rules whose later incumbents and proposal trajectories can diverge.

**Algorithm 1** One SEED-FULL generation

**Require:** incumbent $\theta$; fixed run splits: tune $T$, holdout $H$, canary $C$, final $F$
propose at most $K$ prompt/workflow candidates using visible history
evaluate candidates on $T$ and $H$
discard candidates that worsen tune pass, holdout pass, or holdout gap
**if** no candidate remains **then**
  retain $\theta$ and record a no-op
**else**
  select admitted candidate $e$ using Equation 1
  evaluate $e$ and $\theta$ on the same run-level canary $C$
  **if** $e$ worsens canary pass or widens the canary gap **then**
    retain $\theta$ and record rollback $(\theta, e)$
  **else**
    deploy $e$ as the next incumbent
  **end if**
**end if**
evaluate the realized deployed state on $F$ for reporting only

### 3.2 EVENTS, ADMISSIONS, AND EXACT PAIRS

An *admission event* occurs when a candidate clears the holdout gate; a *first admission* is the first such event in a run. A *first-exact-pair event* is a run's first evaluation of an incumbent–candidate pair; subsequent evaluations are retries. Raw events measure gate load, first admissions characterize initial decisions, and first exact pairs approximate distinct evaluated decisions while preserving the realized trajectory.

The executor receives each canary task statement, but the mutation proposer receives no canary identity, example, outcome, rationale, or score. Nevertheless, rollback retains the incumbent, so later proposals depend on earlier canary decisions. The same canary IDs recur across generations: "fresh"

means separation from tune and holdout, not resampling or independence from earlier deployment decisions. This separation provides no adaptive-valid confidence guarantee.

### 3.3 Estimands and model-cluster uncertainty

For profile $p$, let $A_p$ be the multiset of admitted retry-inclusive events, $U_p$ the subset retaining only the first occurrence of each exact incumbent–candidate pair within a run, and $I_p$ the first admission in each run with at least one admission. The corresponding disagreement rates are

$$r_p^{\text{raw}} = \frac{\sum_{e \in A_p} \mathbf{1}[\text{canary fails}(e)]}{|A_p|}, \qquad r_p^{\text{pair}} = \frac{\sum_{e \in U_p} \mathbf{1}[\text{canary fails}(e)]}{|U_p|}, \tag{2}$$

$$r_p^{\text{first}} = \frac{\sum_{e \in I_p} \mathbf{1}[\text{canary fails}(e)]}{|I_p|}. \tag{3}$$

The raw rate measures rejection per gate exposure; the first-exact rate measures disagreement on a pair's first appearance in the realized run; the first-admission rate measures confirmation failure of the first provisional update. None is a contamination-free probability for the underlying model.

The model-cluster bootstrap follows cluster-resampling principles (Field & Welsh, 2007), with general background in Efron (1992). Each replicate samples model lines with replacement, retaining complete within-line histories and dependence among seeds, families, generations, and repeated pairs. Percentile intervals summarize the empirical panel; they do not make aliases within a provider family independent.

**A realized retry.** The Qwen3.6-Plus Anchor trace, seed 11 (Table 18), illustrates the key. MBPP+ admits and rolls back the same pair in both generations: two exposures, one first-exact decision. On APPS, two candidates pass confirmation; the second becomes the final incumbent. These are illustrative state transitions, not additional panel-wide estimates.

### 3.4 The deployment trace as a reusable record

The reusable deployment trace reconstructs the evidence path rather than reducing an update to its endpoint score. Table 3 lists the minimum candidate-exposure fields; fixed task IDs may be linked from a seed-level manifest instead of duplicated in event rows.

## 4 Experimental Design

**Core panel.** The frozen panel has 24 model lines: nine proprietary hosted-API, five hosted open-weight-family, and ten self-hosted open-weight lines using vLLM (Kwon et al., 2023). HumanEval+, MBPP+, HumanEval Pro, MBPP Pro, ClassEval, and APPS (Liu et al., 2023; Yu et al., 2025; Du et al., 2023; Hendrycks et al., 2021) use five seeds $(11, 29, 47, 61, 83)$; all 24 models, six families, and three profiles are present (432/432 cells). Decoding is deterministic, with local execution and benchmark-specific timeouts. Appendix Tables 10–13 give the roster and task geometry.

**Profiles.** Anchor uses a 6/4/4/6 tune/holdout/canary/final split, two generations, and three proposals per generation. Horizon keeps the 6/4/4/6 split but uses four generations and five proposals; it changes both depth and breadth. Confirmatory uses 10/8/8/10 with two generations and three proposals. Confirmatory therefore changes the entire evidence profile and is not a matched size-only intervention.

**Metrics and uncertainty.** We report hidden pass rate (`HP@1`), visible–hidden gap, admissions, rollback checks, deployments, no-ops, calls, and tokens. Locally *protective* rollbacks reject candidates that would worsen untouched-final pass or gap; direct-pass and gap-only protection are separate. Intervals resample model clusters without treating repeated events as independent. The final split evaluates historical policies, not replacement-gate tuning.

Table 3: Minimum candidate-level deployment trace. The exact-pair key is defined within a run from the incumbent and candidate identities; a repeated key is a retry even when it occurs at a later generation.

| Field group | Required values | Purpose |
|---|---|---|
| Run identity | model line, benchmark family, profile, seed, variant | Defines the matched experimental cell |
| Environment | checkpoint/alias, serving stack, decoding, timeout, artifact version | Makes provider and execution drift visible |
| Task manifest | tune, holdout, canary, and final task IDs plus manifest hash | Verifies disjoint evidence roles |
| Search state | generation, incumbent ID, proposal budget, proposal reason | Reconstructs history-conditioned search |
| Candidate identity | candidate ID, mutation-library member, exact-pair key | Separates unique pairs from retries |
| Tune evidence | pass, gap inputs, visible failure summaries | Records exploratory evidence |
| Admission evidence | holdout pass, holdout gap, hard-gate outcomes, ranking score | Reconstructs provisional selection |
| Confirmation evidence | canary pass/gap for candidate and incumbent | Reconstructs deployment gate |
| Decision | admit, no-op, deploy, or rollback with reason | Records state transition |
| Final evidence | pass and gap for the candidate, incumbent, and deployed state | Supports retrospective local utility |
| Usage | calls, prompt/completion/total tokens, cache status | Quantifies resource footprint |
| Provenance | timestamps, response hashes, evaluator version, invariant status | Supports validation and replay |

Table 4: **One denominator map for the analysis.** The retry-inclusive core is the canonical six-family population; the Fresh-LiveCodeBench extension is reported separately as a low-powered sensitivity. P/G denotes pass-or-gap regression. Exact rates, intervals, and coverage filters are in Appendix C.

| Analysis | Scope | Counts or allocations | Contrast/summary | Interpretation |
|---|---|---|---|---|
| First admission | 24 lines, six families | A/H/C: 145/577; 169/667; 177/527 | 25.1/25.3/33.6% | Initial decision disagreement |
| Retry-inclusive core | Same core panel | A/H/C: 340/959; 918/1,924; 414/851 | H–A $+12.26$ pp | Operational gate burden |
| First exact pair | Same realized runs | A/H: 195/814; 259/1,115 | H–A $-0.73$ pp | Distinct-pair sensitivity |
| Conditional redraw | 10 self-hosted lines | A/H/C events: 317/410/214 | H–A $-0.89$ pp | Fixed-trajectory noise diagnostic |
| Final utility | Replayable rollback subset | 271/652 P/G; 159/652 pass | Recall 33.9/33.3% | First-exact local protection |
| Fixed budget | 57/60 self-hosted lines | 5/0, 4/1, 3/2, 2/3 task allocations | All endpoint CIs cross 0 | Allocation sensitivity |

# 5 Results

## 5.1 Canary confirmation disagrees with provisional admission

At first admission, the disjoint canary rejects 145/577 (25.1%) Anchor, 169/667 (25.3%) Horizon, and 177/527 (33.6%) Confirmatory decisions. Thus a second disjoint check changes the deployment decision for roughly one quarter of initial Anchor/Horizon admissions and one third of Confirmatory admissions. This finding supports the procedural claim that hidden admission is provisional once it participates in selection.

Raw events tell a different operational story. Six-family counts are 340/959 (35.5%), 918/1,924 (47.7%), and 414/851 (48.6%). The Anchor–Horizon difference is $+12.26$ percentage points with a 95% model-cluster interval of $[9.64, 15.04]$. The raw statistic is not a distinct-decision effect: exact-pair repeats account for 145/382 (38.0%) of admitted Anchor generation-2 events, 169/423 (40.0%), 232/417 (55.6%), and 408/417 (97.8%) of Horizon generations 2–4, and 177/324 (54.6%) of Confirmatory generation-2 events.

Keeping only the first exact pair per run yields 195/814 for Anchor and 259/1,115 for Horizon, a difference of $-0.73$ points ($[-2.37, 0.91]$). The raw-event NO ROLLBACK contrast is similarly flat at $-0.15$ points ($[-2.14, 1.83]$). The first-exact-pair interval crossing zero does not support a distinct-pair effect (Table 5). Anchor and Horizon differ in both depth and breadth, and the large raw contrast is generated by repeated gate exposures along a history-conditioned trajectory.

Table 5: Retry-aware Anchor–Horizon disagreement contrasts. Intervals resample model clusters. †The incomplete blacklist replay has 154/180 fully completed five-seed cells; the contrast uses 275 matched Anchor/Horizon seed-run pairs (coverage accounting in Appendix E).

| Analysis unit | Anchor | Horizon | H–A pp [95% interval] |
|---|---|---|---|
| Retry-inclusive SEED-FULL events | 340/959 | 918/1,924 | $+12.26$ $[+9.64, +15.04]$ |
| First exact pair | 195/814 | 259/1,115 | $-0.73$ $[-2.37, +0.91]$ |
| NO ROLLBACK, raw events | 209/882 | 388/1,648 | $-0.15$ $[-2.14, +1.83]$ |
| Failed-pair blacklist† | 78/315 | 136/513 | $+1.75$ $[-1.27, +6.14]$ |

### 5.2 A CONDITIONAL REPARTITION DIAGNOSTIC DOES NOT RECOVER THE RAW GAP

On the ten-line self-hosted final-evidence subset, all 1,375 retained events have complete task-level outcomes and pass the stored-outcome invariants. The first-exact subset contains 317/410/214 Anchor/Horizon/Confirmatory events. Using all admitted first-exact events, the observed Anchor–Horizon disagreement difference is $-0.89$ points ($[-3.70, 1.77]$); conditional repartition gives $-0.39$ ($[-2.01, 0.90]$), and the difference-in-differences is $-0.50$ ($[-1.86, 1.07]$). Restricting to 235 common-admission cells leads to the same qualitative conclusion (Appendix D).

This diagnostic holds the realized incumbent, candidate, admission, and future trajectory fixed while repartitioning retained outcomes. It is not a trajectory-level permutation test, and it cannot identify the effect of rollback on future proposals. It does show that the large retry-inclusive Anchor–Horizon contrast does not persist at the distinct-pair unit on the complete retained-outcome surface.

### 5.3 FAILED-PAIR BLACKLISTING CHANGES THE PATH BUT NOT THE EVIDENTIAL BOUNDARY

As a trajectory-level sensitivity, we blacklist an exact pair after its first canary failure. Filtering occurs before proposal-budget truncation, allowing the next eligible proposal to backfill the budget. Previously accepted pairs may still recur; this is not a globally unique-decision policy.

The replay has 154/180 fully completed self-hosted model–family–profile cells, each with five planned seed-runs, and 870/900 completed seed-runs: $154 \times 5 = 770$ from full cells plus 100 from 26 partial cells gives $770 + 100 = 870$. The 30 unfinished seed-runs stopped at missing frozen response files (Appendix E). Exact joins across model, benchmark family, and seed produce 275 matched Anchor/Horizon run pairs, 60 lines, and ten models. Rollbacks/admissions are 78/315 versus 136/513, for a $+1.75$ point per-admission difference with interval $[-1.27, 6.14]$. The 25 unavailable Horizon branches count unmatched seed-runs, not the 26 partial cells. Their observed Anchor histories contain 35 rollbacks among 48 admissions (72.92%); the missing branches remain outcome-informative.

The positive rollbacks-per-run bound is partly mechanical because Horizon is both deeper and wider. Together with incomplete outcome coverage, this prevents the blacklist replay from supporting a distinct-decision causal claim.

Table 6: Missing-outcome bounds for the failed-pair-blacklist sensitivity. The ranges are identification bounds over the full target of 300 Anchor/Horizon seed-run pairs (60 model–family lines $\times$ five seeds), not confidence intervals for the 275 observed pairs.

| Quantity | Full-run H–A identification range |
|---|---|
| Rollback probability per admission | $[-8.94, +7.37]$ pp |
| Probability of any rollback per run | $[-1.33, +7.00]$ pp |
| Rollbacks per run | $[+0.0767, +0.4100]$ |

### 5.4 Rollback prevents some local final-split regressions

Table 7 reports both counting units. Among 1,579 retry-inclusive rollbacks with final evidence, 654 (41.4%) reject a candidate that would worsen final pass or gap, and 350 (22.2%) reject one that would directly reduce final pass. At the first-exact-pair unit, 271/652 (41.6%) rollbacks prevent a pass-or-gap regression and 159/652 (24.4%) prevent direct pass loss, close to the retry-inclusive precision. The 159 direct-pass events occur in 159 distinct runs and sum to 225 decision-by-task final-loss counterfactuals: 110 decisions lose one final task, 35 lose two, 11 lose three, and three lose four.

Precision alone can exaggerate usefulness, so we also report recall. The first-exact canary catches 271/799 (33.9%) pass-or-gap regressions and 159/478 (33.3%) direct-pass regressions. It therefore misses about two thirds of the local regressions defined by these metrics and unnecessarily rejects many candidates that would not regress on the small final slice. We call these *local final regressions*; they are not measurements of downstream user outcomes. Selecting a new threshold from their operating curve would require another untouched policy-selection split.

Table 7: Local untouched-final utility under two analysis units. "Runs" is the number of distinct runs containing the 159 first-exact direct-pass cases; the retry-inclusive row shares that distinct-run support.

| Unit | Checks | P/G protect | Pass protect | Runs | Task-loss sum |
|---|---|---|---|---|---|
| Retry-inclusive | 1,579 | 654 (41.4%) | 350 (22.2%) | 159 | 474 |
| First exact pair | 652 | 271 (41.6%) | 159 (24.4%) | 159 | 225 |

### 5.5 A fixed task budget exposes trade-offs, not an optimum

We replay a five-task holdout/canary budget on 57/60 model–family lines across ten self-hosted models. Moving from a 4/1 to a 2/3 allocation increases provisional admissions (348 to 413), rollback checks (67 to 145), and the absolute number of protective rollbacks (26 to 45), while decreasing deployed updates (281 to 268), pass-or-gap precision (38.81% to 31.03%), mean calls (87.7 to 78.4), and mean tokens (105.8k to 93.9k). The intermediate 3/2 allocation lies between these operational profiles (Table 8).

Relative to a 5/0 hidden-only allocation, final-`HP@1` deltas are $+0.57$ points ($[-0.81, 1.89]$), $+0.48$ ($[-0.82, 1.76]$), and $+0.29$ ($[-1.14, 1.70]$) for 4/1, 3/2, and 2/3. Allocation-matched SEED-Full versus No Rollback intervals also cross zero (Table 9). These data do not identify an endpoint-optimal split or establish equivalence. They expose a policy trade-off among admissions, checks, deployments, compute, and local protection.

### 5.6 External checks remain limited

A matched Fresh-LiveCodeBench (Jain et al., 2025) Anchor/Horizon extension covers four hosted Qwen family lines. Its raw pooled contrast is $+25.16$ points ($[9.29, 38.37]$), but the exact model sign test has $p = 0.125$; first-exact Horizon–Anchor is $+8.47$ points ($[-8.97, 21.67]$) with $p = 1$. Protective utility is 4/21 raw and 1/6 first exact. The extension is a low-powered freshness sensitivity, not evidence that the core benchmarks are uncontaminated.

Table 8: Operational trade-off for SEED-FULL. Calls and tokens are realized means per run; precision is pass-or-gap.

| Alloc. | Admit | Deploy | Check | Protect | P/G prec. | Calls | Tokens |
|---|---|---|---|---|---|---|---|
| 4/1 | 348 | 281 | 67 | 26 | 38.81% | 87.7 | 105.8k |
| 3/2 | 391 | 276 | 115 | 36 | 31.30% | 83.1 | 100.1k |
| 2/3 | 413 | 268 | 145 | 45 | 31.03% | 78.4 | 93.9k |

Table 9: Final-`HP@1` contrasts in percentage points. All model-bootstrap intervals cross zero.

| Allocation | Versus 5/0 hidden-only | Versus matched NO ROLLBACK |
|---|---|---|
| 4/1 | $+0.57\ [-0.81, +1.89]$ | $+0.34\ [-0.38, +1.12]$ |
| 3/2 | $+0.48\ [-0.82, +1.76]$ | $+0.22\ [-0.66, +1.16]$ |
| 2/3 | $+0.29\ [-1.14, +1.70]$ | $+0.11\ [-0.69, +1.01]$ |

The floor-saturated 23-line SWE-bench Lite pilot (Jimenez et al., 2024) on `sympy/sympy` (Meurer et al., 2017) illustrates abstention and no-ops without hidden lift, not repository-scale rollback precision. No non-coding agent experiment was run; both repository-scale precision and non-coding transfer remain open.

## 6 DISCUSSION

**What the data support.** The result is procedural and diagnostic: selection-exposed hidden admission is provisional, and a disjoint check changes many initial deployment decisions in this bounded setting. The trace separates initial admissions, repeated exposures, rollbacks, no-ops, local final regressions, and endpoints. Removing retries eliminates the apparent search-intensity effect while leaving local rollback utility similar across counting units. First-exact filtering retains the historical trajectory; it is not a globally unique-decision policy. The separate failed-pair-blacklist sensitivity changes the path but remains limited by incomplete coverage (Section 5.3).

**What the data do not support.** The study establishes neither increased distinct non-transfer from deeper search, fixed-budget endpoint gains from canarying, nor an optimal holdout/canary allocation. It does not detect training-data contamination, isolate split size from other profile changes, or establish repository-scale or non-coding transfer. The proposal-channel-isolated but system-adaptive canary influences later proposals through rollback.

**Cost.** The protocol is expensive: comparable hosted Anchor/Confirmatory slices use about 104,133 calls and 231.6M tokens for SEED-FULL, versus 8,765 calls and 18.2M tokens for REFLECTION-ONLY. Prices, aliases, and cache policies change, precluding stable dollar costs. More canary tasks can reduce calls through earlier rejection: fixed-budget cost reflects policy, not an isolated mechanism.

**Implications for reporting.** Report proposal, admission, confirmation, and reporting data; rollback's effects on future proposals; separate retry-inclusive and unique-decision counts; and protective-gate precision and recall. Retain null and negative results: zero-crossing endpoint intervals do not justify a universal budget split.

## 7 CONCLUSION

SEED provides an executable, retry-aware deployment trace separating admission and deployment evidence for bounded adaptive coding-agent updates. Canary confirmation rejects roughly a quarter of first Anchor/Horizon admissions and a third of Confirmatory admissions. The large raw Anchor–Horizon gap reflects repeated gate exposures and vanishes under first-exact-pair counting. Rollback catches some concrete local final-split regressions but misses many; fixed-budget endpoints remain inconclusive. These empirical findings do not establish that disjoint canaries solve contamination or define a universally optimal update policy.

## AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental protocols, assist with implementing the methods and analysis code, support consistency and lineage checks, and help interpret results. Additionally, we used these tools to retrieve, identify, and organize relevant literature, format references, prepare research artifacts, draft sections of the manuscript, and edit the writing for clarity. Formal proof construction and proof writing are not applicable to the work reported here, which presents an empirical protocol and analysis rather than formal theorems.

Separately, the hosted and self-hosted language models evaluated in this study generated candidate programs for existing benchmark tasks under the bounded prompt/workflow configurations. These programs are evaluated system outputs, rather than newly synthesized benchmark tasks or reference labels. Benchmark execution and harness-defined rules, not free-form AI judgments, supplied the reported scores and admission, deployment, or rollback decisions (Section 3 and Appendices A and B).

We have reviewed all AI-assisted work, including the implementation, model and data provenance, experimental outputs, statistical analyses, citations, figures, and manuscript claims. Automated evaluation did not replace author review. We take responsibility for the final content of this work, including all text, code, analyses, claims, and artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

The anonymous review artifact is available at https://anonymous.4open.science/r/seed-review-artifact-D056. Section 3, Appendix A, and Algorithm 1 specify the four evidence roles, mutation library, ranking rule, and admission and rollback gates. Section 4 and Appendix B document the model roster, serving paths, benchmark task geometry, shared split manifests, seeds, decoding, and execution timeouts. Appendix B.2 records the unresolved historical checkpoint for the Mistral alias. Table 4 and Appendix C define the analysis populations, retry-aware counting units, and model-cluster uncertainty estimates. Appendices D–G provide the conditional repartition, trajectory-sensitivity, local-utility, and fixed-budget analyses; Appendices H and I document external-check coverage and resource-accounting scope. Appendix J describes the anonymized harness, benchmark adapters, mutation library, split manifests, cached model responses where policy permits, event records, aggregation scripts, and figure/table exporters. Appendix K specifies the trace fields needed to reconstruct the evidence path and distinguish retry-inclusive events, first admissions, and first exact pairs. Hosted-system drift and other limits to replication are discussed in Appendix L.

## ETHICS STATEMENT

SEED studies bounded prompt/workflow updates on coding benchmarks, not unrestricted self-modification or a validated production safety policy. Functional test outcomes, pass-rate gaps, and local final-split regressions do not directly measure exploitability, privacy, maintainability, human acceptance, or downstream user outcomes. Admission or canary success should therefore not be interpreted as a safety certificate. Rollback can both miss regressions and reject candidates that would not regress on the evaluated final slice. Although canary feedback is excluded from mutation-proposal history, its effect on the surviving incumbent makes the overall system adaptive. We therefore distinguish retry-inclusive gate burden from distinct-pair evidence, retain null and negative results, and make no claim of contamination-free evaluation or transfer beyond the evaluated coding settings. Any deployment policy selected using the reported final-split outcomes requires prospective validation on a new untouched evaluation set; those same outcomes cannot also provide independent validation of the selected policy. Resource costs and the boundaries of responsible interpretation are discussed in Section 6 and Appendices I and L.

## REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 Technical Report. *arXiv preprint arXiv:2412.08905*, 2024. URL https://arxiv.org/abs/2412.08905.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925*, 2025. URL https://arxiv.org/abs/2508.10925.

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, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Alibaba Cloud. Alibaba Cloud Model Studio: Recommended Models. Alibaba Cloud Documentation, 2026. URL https://www.alibabacloud.com/help/en/model-studio/models. Accessed: 2026-04-10.

Anonymous. HarnessOpt-Bench: Evaluating LLMs at harness optimization. Submitted to ACL Rolling Review, August 2026, 2026. URL https://openreview.net/forum?id=eBSxqRO1Go. Under review.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. URL https://arxiv.org/abs/1907.02893.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021. URL https://arxiv.org/abs/2112.00861.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a. URL https://arxiv.org/abs/2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*, 2022b. URL https://arxiv.org/abs/2212.08073.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Albert Bifet, Ricard Gavaldà, Geoffrey Holmes, and Bernhard Pfahringer. *Machine Learning for Data Streams: With Practical Examples in MOA*. MIT Press, 2018.

Avrim Blum and Moritz Hardt. The Ladder: A Reliable Leaderboard for Machine Learning Competitions. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1006–1014. PMLR, 2015. URL https://proceedings.mlr.press/v37/blum15.html.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. URL https://arxiv.org/abs/2108.07258.

Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D Sculley. The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction. In *2017 IEEE international conference on big data (big data)*, pp. 1123–1132. IEEE, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models Are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering Latent Knowledge in Language Models Without Supervision. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. InternLM2 Technical Report. *arXiv preprint arXiv:2403.17297*, 2024. URL https://arxiv.org/abs/2403.17297.

Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107, 2010.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

CodeGeeX Team. CodeGeeX4-ALL-9B. Hugging Face model card, 2024. URL https://huggingface.co/zai-org/codegeex4-all-9b. Accessed: 2026-09-26.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. *Journal of Machine Learning Research*, 23(226):1–61, 2022.

DeepSeek. DeepSeek-V3.2 Release. DeepSeek API Documentation, dec 2025. URL https://api-docs.deepseek.com/news/news251201/. Accessed: 2026-04-10.

DeepSeek-AI. DeepSeek-V3.2. Hugging Face model card, 2025. URL https://huggingface.co/deepseek-ai/DeepSeek-V3.2. Accessed: 2026-04-10.

Mostafa Dehghani, Yi Tay, Alexey A Gritsenko, Zhe Zhao, Neil Houlsby, Fernando Diaz, Donald Metzler, and Oriol Vinyals. The benchmark lottery. *arXiv preprint arXiv:2107.07002*, 2021. URL https://arxiv.org/abs/2107.07002.

Sylvie Delacroix. Mind the Gap: Machine Learning Systems and the Passage of Time. The Alan Turing Institute Blog, apr 2022. URL https://www.turing.ac.uk/blog/mind-gap-machine-learning-systems-and-passage-time.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating Data Contamination in Modern Benchmarks for Large Language Models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8706–8719, Mexico City, Mexico, jun 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.482. URL https://aclanthology.org/2024.naacl-long.482/.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=kiYqbO3wqw.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI Agents Solve Long-Horizon Software Engineering Tasks? *arXiv preprint arXiv:2509.16941*, 2025. URL https://arxiv.org/abs/2509.16941v2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 4171–4186, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-Level Code Generation. *arXiv preprint arXiv:2308.01861*, 2023. URL https://arxiv.org/abs/2308.01861.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, volume 28, 2015a. URL https://arxiv.org/abs/1506.02629.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015b. URL https://research.ibm.com/publications/the-reusable-holdout-preserving-validity-in-adaptive-data-analysis.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pp. 569–593. Springer, 1992.

Christopher A Field and Alan H Welsh. Bootstrapping clustered data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(3):369–390, 2007.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059. PMLR, 2016. URL https://proceedings.mlr.press/v48/gal16.html.

Balaji Dinesh Gangireddi, Aniketh Garikaparthi, Manasi Patwardhan, and Arman Cohan. Self-Adapting Agents for Automating Research Coding Workflows. In *ICLR 2026 Workshop on Recursive Self-Improvement*, 2026. URL https://openreview.net/forum?id=ihcRmUkXHF.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022. URL https://arxiv.org/abs/2209.07858.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding Code LLMs in Execution Feedback with Reinforcement Learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19034–19055. PMLR, 2025. URL https://proceedings.mlr.press/v267/gehring25a.html.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. URL https://arxiv.org/abs/2007.01434.

Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hkg4TI9xl.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring Coding Challenge Competence with APPS. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://openreview.net/forum?id=sD93GOzH3i5.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Evan Hubinger, Chris Van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. Risks from learned optimization in advanced machine learning systems. *arXiv preprint arXiv:1906.01820*, 2019. URL https://arxiv.org/abs/1906.01820.

Nurit Cohen Inger, Yehonatan Elisha, Bracha Shapira, Lior Rokach, and Seffi Cohen. Forget What You Know about LLMs Evaluations - LLMs are Like a Chameleon. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21664–21677, Suzhou, China, nov 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1098. URL https://aclanthology.org/2025.emnlp-main.1098/.

InternLM Team. InternLM3-8B-Instruct. Hugging Face model card and InternLM documentation, 2025. URL https://huggingface.co/internlm/internlm3-8b-instruct. See also https://internlm.readthedocs.io/en/latest/model_card/InternLM3.html. Accessed: 2026-05-06.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018. URL https://arxiv.org/abs/1805.00899.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination-Free Evaluation of Large Language Models for Code. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Sihang Jiang, Lipeng Ma, Zhonghua Hong, Keyi Wang, Zhiyu Lu, Tengfei Wang, Shisong Chen, Jinghao Zhang, Tianjun Pan, Weijia Li, Jiaqing Liang, and Yanghua Xiao. SEA-Eval: A Benchmark for Evaluating Self-Evolving Agents Beyond Episodic Assessment. *arXiv preprint arXiv:2604.08988*, 2026. URL https://arxiv.org/abs/2604.08988.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Yiyang Jin, Kunzhao Xu, Hang Li, Xueting Han, Yanmin Zhou, Cheng Li, and Jing Bai. ReVeal: Self-Evolving Code Agents via Reliable Self-Verification. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=q56ZI1Co43.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. URL https://arxiv.org/abs/2207.05221.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022. URL https://arxiv.org/abs/2210.02406.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking Benchmarking in NLP. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124, Online, jun 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.324. URL https://aclanthology.org/2021.naacl-main.324/.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114 (13):3521–3526, 2017.

Rafal Kocielnik, Saleema Amershi, and Paul N Bennett. Will you accept an imperfect ai? exploring designs for adjusting end-user expectations of ai systems. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–14, 2019.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. WILDS: A Benchmark of in-the-Wild Distribution Shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664. PMLR, 2021. URL https://proceedings.mlr.press/v139/koh21a.html.

Holger Krekel, Bruno Oliveira, Ronny Pfannschmidt, Floris Bruynooghe, Brianna Laugher, and Freya Bruhin. pytest. Software, 2004. URL https://github.com/pytest-dev/pytest. Project citation; a specific version is not asserted.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018. URL https://arxiv.org/abs/1811.07871.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=3IyL2XWDkG.

Percy Liang et al. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=iO4LZibEqW. Featured Certification, Expert Certification, Outstanding Certification.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023. URL https://arxiv.org/abs/2305.01210.

Siwei Liu, Jinyuan Fang, Han Zhou, Yingxu Wang, and Zaiqiao Meng. SEW: Self-Evolving Agentic Workflows for Automated Code Generation. *arXiv preprint arXiv:2505.18646*, 2025. URL https://arxiv.org/abs/2505.18646.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Inbal Magar and Roy Schwartz. Data Contamination: From Memorization to Exploitation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 157–165, Dublin, Ireland, may 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.18. URL https://aclanthology.org/2022.acl-short.18/.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. SymPy: Symbolic Computing in Python. *PeerJ Computer Science*, 3:e103, 2017.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented Language Models: A Survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=jh7wH2AzKK. Survey Certification.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A Benchmark for General AI Assistants. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Mistral AI. Mistral-Small-24B-Instruct-2501. Hugging Face model card, 2025a. URL https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501. Accessed: 2026-09-26.

Mistral AI. Model Card for Mistral-Small-3.1-24B-Instruct-2503. Hugging Face model card, 2025b. URL https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Instruct-2503. Accessed: 2026-04-10.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pp. 220–229, 2019.

NIST/SEMATECH. Measures of Scale. NIST/SEMATECH e-Handbook of Statistical Methods, n.d. URL https://www.itl.nist.gov/div898/handbook/eda/section3/eda356.htm. Accessed: 2026-04-11.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red Teaming Language Models with Language Models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448, Abu Dhabi, United Arab Emirates, dec 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.225. URL https://aclanthology.org/2022.emnlp-main.225/.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164):1–20, 2021. URL https://jmlr.org/papers/v22/20-303.html.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Xue Qin, Simin Luan, John See, Zeyd Boukhers, Cong Yang, and Zhijun Li. Governed Capability Evolution: Lifecycle-Time Compatibility Checking and Rollback for AI-Component-Based Systems, with Embodied Agents as Case Study. *arXiv preprint arXiv:2604.08059*, 2026. URL https://arxiv.org/abs/2604.08059v5.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. Tool Learning with Foundation Models. *ACM Computing Surveys*, 57(4):1–40, 2024. doi: 10.1145/3704435. URL https://doi.org/10.1145/3704435.

Qwen Team. QwQ-32B. Hugging Face model card and Qwen blog, 2025a. URL https://huggingface.co/Qwen/QwQ-32B. See also https://qwenlm.github.io/blog/qwq-32b/. Accessed: 2026-05-06.

Qwen Team. Qwen3-Next-80B-A3B-Instruct. Hugging Face model card, sep 2025b. URL https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct. Accessed: 2026-04-10.

Qwen Team. Qwen3-Next-80B-A3B-Thinking. Hugging Face model card, 2025c. URL https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Thinking. Accessed: 2026-09-26.

Qwen Team. Qwen3.6-Plus: Towards Real World Agents. Qwen Blog, apr 2026a. URL https://qwen.ai/blog?id=qwen3.6. Accessed: 2026-04-10.

Qwen Team. Qwen3-Coder-Next. Hugging Face model card, feb 2026b. URL https://huggingface.co/Qwen/Qwen3-Coder-Next. Accessed: 2026-04-10.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet Classifiers Generalize to ImageNet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, Online, jul 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025. URL https://arxiv.org/abs/2504.15228.

Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4):105–114, 2015.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022. URL https://arxiv.org/abs/2206.05802.

Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018. doi: 10.14778/3229863.3229867.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Shuai Shao, Qihan Ren, Dongrui Liu, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, and Jing Shao. Your Agent May Misevolve: Emergent Risks in Self-Evolving LLM Agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a24cd16bc361afa78e57d31d34f3d936-Abstract-Conference.html.

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, pp. 8634–8652, 2023.

Ali Shirali, Rediet Abebe, and Moritz Hardt. A Theory of Dynamic Benchmarks. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=i8L9qoeZOS.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Aarohi Srivastava et al. Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj. Featured Certification.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021, 2020.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Harini Suresh and John Guttag. A framework for understanding sources of harm throughout the machine learning life cycle. In *Proceedings of the 1st ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, pp. 1–9, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998. ISBN 9780262193986. URL https://mitpress.mit.edu/9780262193986/reinforcement-learning/.

SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 2024. URL https://arxiv.org/abs/2401.01313.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971.

Varun Ursekar, Apaar Shanker, Veronica Chatrath, Yuan Xue, and Samuel Marc Denton. VeRO: A Harness for Agents to Optimize Agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=zQzmwG2Nue.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. URL https://arxiv.org/abs/2006.10726.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Wenyi Wang, Piotr Piękos, Nanbo Li, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel Machine: Human-Level Coding Agent Development by an Approximation of the Optimal Self-Improving Machine. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T0EiEuhOOL.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain-of-Thought Reasoning in Language Models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Alec Warner and Štěpán Davidovič. Canarying Releases. In Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, and Stephen Thorne (eds.), *The Site Reliability Workbook: Practical Ways to Implement SRE*, chapter 16. O'Reilly Media, 2018. URL https://sre.google/workbook/canarying-releases/. With Alex Hidalgo, Betsy Beyer, Kyle Smith, and Matt Duftler.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How Does LLM Safety Training Fail? In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=jA235JGM09.

Lilian Weng. LLM-Powered Autonomous Agents. Lil'Log, Jun 2023. URL https://lilianweng.github.io/posts/2023-06-23-agent/.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

Boming Xia, Qinghua Lu, Liming Zhu, Zhenchang Xing, Dehai Zhao, and Hao Zhang. Evaluation-Driven Development and Operations of LLM Agents: A Process Model and Reference Architecture. *arXiv preprint arXiv:2411.13768*, 2024. URL https://arxiv.org/abs/2411.13768.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Linyi Yang, Yaoxian Song, Xuan Ren, Chenyang Lyu, Yidong Wang, Jingming Zhuo, Lingqiao Liu, Jindong Wang, Jennifer Foster, and Yue Zhang. Out-of-Distribution Generalization in Natural Language Processing: Past, Present, and Future. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4533–4559, Singapore, dec 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.276. URL https://aclanthology.org/2023.emnlp-main.276/.

Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, Yibo Miao, Bofei Gao, Yejie Wang, Yingwei Ma, Yanhao Li, Yue Liu, Zhenxing Hu, Kaitai Zhang, Shuyi Wang, Huarong Chen, Hongyong Song, Yang Liu, Yang Gao, Zhilin Yang, and Tianyu Liu. Kimi-Dev: Agentless Training as Skill Prior for SWE-Agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f039544a51899ebac128dbec33dae81c-Abstract-Conference.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R. Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Zhaojian Yu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. HumanEval Pro and MBPP Pro: Evaluating Large Language Models on Self-invoking Code Generation Task. In Wanxiang Che, Joyce

Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13253–13279, Vienna, Austria, jul 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.686. URL https://aclanthology.org/2025.findings-acl.686/.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *arXiv preprint arXiv:2508.06471*, 2025. URL https://arxiv.org/abs/2508.06471.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pUpzQZTvGY.

Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, Elsie Nallipogu, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, and Dongmei Zhang. SWE-bench Goes Live! In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d83c4a745789690f82e86d0ef752ae7c-Abstract-Datasets_and_Benchmarks_Track.html.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, Xiaolei Wang, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Yiwen Hu, Jian-Yun Nie, and Ji-Rong Wen. A Survey of Large Language Models. *Frontiers of Computer Science*, 20 (12):2012627, 2026. doi: 10.1007/s11704-026-60308-3. URL https://doi.org/10.1007/s11704-026-60308-3.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, et al. CodeGeeX: A Pre-Trained Model for Code Generation with Multilingual Benchmarking on HumanEval-X. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 5673–5684, 2023.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence. *arXiv preprint arXiv:2406.11931*, 2024. URL https://arxiv.org/abs/2406.11931.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. URL https://arxiv.org/abs/1909.08593.

Martin Zinkevich. Rules of Machine Learning: Best Practices for ML Engineering. Google Developers, 2017. URL https://developers.google.com/machine-learning/guides/rules-of-ml.

# APPENDIX

## A PROTOCOL AND IMPLEMENTATION DETAILS

This appendix defines the implementation, evidence roles, analysis units, and coverage behind the reported quantities.

### A.1 DEPLOYMENT PROCEDURE

Each run fixes a model line, benchmark family, profile, seed, and task manifest. The incumbent begins with `base_direct`. The proposal rule uses visible failures and the incumbent's recent history to prioritize members of the bounded mutation library. The executor receives each task statement when the candidate is evaluated, but holdout, canary, and final task identities, examples, outcomes, and scores never enter mutation-proposal history.

Algorithm 1 and the hard-gate definitions are in Section 3. The score-coefficient discussion below records implementation choices, not a fitted optimum.

**Fixed task partitions.** At run initialization, the split routine sorts the retained problem IDs, shuffles them once with the run seed, and cuts consecutive tune, holdout, canary, and final blocks of the profile-specified sizes. The generation loop reuses those lists without resampling. For the same benchmark pool and seed, Anchor and Horizon share the 6/4/4/6 partition; Confirmatory's 10/8/8/10 allocation changes the block boundaries but remains fixed within its run. The recorded Mistral manifests agree on Anchor/Horizon canary IDs for MBPP+, HumanEval+, ClassEval, and APPS. For example, on MBPP+ seed 11, both profiles use `Mbpp/66`, `Mbpp/120`, `Mbpp/8`, and `Mbpp/767`. Canary IDs are stored in the seed-level split list; event logs record canary scores rather than a second copy of those IDs. The seed-level list together with its reuse in the generation loop establishes the within-run task identity.

**Score coefficients.** The $(0.4, 0.6, 0.75)$ visible/hidden/gap weights are fixed globally. A visible-only score would optimize the most exposed evidence; a hidden-only score would concentrate selection pressure on the provisional holdout; removing the gap penalty can favor candidates that widen the visible–hidden discrepancy. The chosen coefficients encode a transparent prior rather than a fitted optimum. Nearby-rule replay is available only for one fully packaged model line and is therefore a mechanism check, not panel-wide robustness evidence.

### A.2 MUTATION LIBRARY AND VARIANTS

Table 2 in the main text lists every library member.

Frozen never mutates. REFLECTION-ONLY uses a static self-review configuration. Unconstrained search ranks with visible evidence. SINGLE EVALUATOR adds hidden admission but no canary confirmation. NO ROLLBACK uses the same selection rule as SEED-FULL but deploys canary-failed candidates; later selection paths can diverge. SEED-FULL rolls them back, while SEED-TIGHT adds stricter evidence margins. Anchor and Confirmatory use the seven-variant lattice; Horizon retains the four sequential-decision variants whose state can diverge over multiple generations.

## B PANEL COMPOSITION AND EXECUTION CONTROLS

### B.1 MODEL ROSTER

The roster is partitioned by the access path used in this study rather than by ambiguous release-status labels. The first two rows of Table 10 form the 14-line hosted-access accounting slice; the second and third rows together form a 15-line open-weight-family diagnostic that overlaps the accounting partition and is not used as the primary roster.

Table 10: Non-overlapping access taxonomy for the 24 model lines. Hosted aliases are named as evaluated; the exact served weights of a provider alias may be undisclosed or change outside a dated snapshot.

| Access bucket | Lines | Model lines |
|---|---|---|
| Proprietary hosted API | 9 | qwen-turbo; qwen-plus; qwen-max; qwen-coder-plus; qwen3-coder-plus; qwen3.6-plus-2026-04-02; qwen3-max-2026-01-23; qwen3-coder-flash; qwen3.5-flash-2026-02-23 |
| Hosted open-weight-family | 5 | deepseek-v3.2; qwen3-coder-next; qwen3-next-80b-a3b-instruct; qwen3-next-80b-a3b-thinking; qwq-plus-2025-03-05 |
| Self-hosted open-weight vLLM | 10 | llama-3.1-8b-instruct; gpt-oss-20b; deepseek-coder-v2-lite-instruct; gemma-3-27b-it; phi-4; codegeex4-9b; mistral-small-24b; internlm3-8b-instruct; kimi-dev-72b; glm-4.5-air |

### B.2 Per-line metadata

Tables 11 and 12 retain the model-line metadata needed to interpret access path and public scale. "Hosted open-weight-family" describes the model family; it does not assert that a provider alias exposes the exact released checkpoint. Provider documentation describes the hosted catalogue (Alibaba Cloud, 2026; Qwen Team, 2026a); the Qwen3 report (Yang et al., 2025) is family background, not proof of an alias's weights. DeepSeek's release documentation and card (DeepSeek, 2025; DeepSeek-AI, 2025), and the individual Qwen3-Coder-Next and Qwen3-Next cards (Qwen Team, 2026b; 2025b;c), document the corresponding released families. The QwQ-32B card (Qwen Team, 2025a) provides family context only for `qwq-plus-2025-03-05`; it does not establish checkpoint equivalence.

**Public parameter counts.** The 685.4B and 72.7B entries are rounded Hugging Face safetensors totals for DeepSeek-V3.2 and Kimi-Dev-72B: 685,355,329,792 and 72,706,203,648 parameters, respectively (DeepSeek-AI, 2025; **?**). These are public-release counts, not evidence identifying the weights served behind the hosted `deepseek-v3.2` alias. Kimi was self-hosted, but its 72.7B entry is likewise public-release metadata, not a recount of the loaded file; "72B" is the release name.

Table 11: Metadata for the 14 hosted-access model lines. Public scale is listed only when disclosed for the corresponding family.

| Model line | Family/access | Public scale | Note |
|---|---|---|---|
| qwen-turbo | proprietary hosted | undisclosed | Fast general-purpose provider alias |
| qwen-plus | proprietary hosted | undisclosed | Mid-tier general-purpose provider alias |
| qwen-max | proprietary hosted | undisclosed | Higher-capability general provider alias |
| qwen-coder-plus | proprietary hosted coder | undisclosed | Earlier hosted coding line |
| qwen3-coder-plus | proprietary hosted coder | undisclosed | Coding-agent provider alias |
| qwen3.6-plus-2026-04-02 | proprietary hosted | undisclosed | Dated snapshot with the deepest packaged diagnostic |
| qwen3-max-2026-01-23 | proprietary hosted | undisclosed | Dated general-model snapshot |
| qwen3-coder-flash | proprietary hosted coder | undisclosed | Lower-cost coding alias |
| qwen3.5-flash-2026-02-23 | proprietary hosted | undisclosed | Dated lower-cost snapshot |
| deepseek-v3.2 | hosted open-weight family | 685.4B public total | API-accessed reasoning/agent model |
| qwen3-coder-next | hosted open-weight family | 80B / 3B active | API-accessed coding-agent model |
| qwen3-next-80b-a3b-instruct | hosted open-weight family | 80B / 3B active | API-accessed instruct line |
| qwen3-next-80b-a3b-thinking | hosted open-weight family | 80B / 3B active | API-accessed thinking line |
| qwq-plus-2025-03-05 | open-family hosted snapshot | undisclosed | Exact served SKU weights not separately exposed |

The self-hosted families are documented by the Llama 3 report (Grattafiori et al., 2024), the gpt-oss model card (Agarwal et al., 2025), the DeepSeek-Coder-V2, Gemma 3, Phi-4, and GLM-4.5 reports (Zhu et al., 2024; Gemma Team et al., 2025; Abdin et al., 2024; Zeng et al., 2025), and the Kimi-Dev paper (Yang et al., 2026). CodeGeeX4 and InternLM3 require their own generation-specific cards (CodeGeeX Team, 2024; InternLM Team, 2025).

Table 12: Metadata for the ten self-hosted vLLM model lines. The serving accommodations change scheduling, not weights, prompts, splits, or scoring.

| Model line | Family | Public scale | Serving note |
|---|---|---|---|
| llama-3.1-8b-instruct | Llama 3.1 | 8B | Standard self-hosted instruct line |
| gpt-oss-20b | GPT-OSS | 21B / 3.6B active | Self-hosted MoE release |
| deepseek-coder-v2-lite-instruct | DeepSeek-Coder-V2 | 16B / 2.4B active | Self-hosted coder MoE |
| gemma-3-27b-it | Gemma 3 | 27B | Self-hosted instruction model |
| phi-4 | Phi-4 | 14B | Self-hosted reasoning-oriented SLM |
| codegeex4-9b | CodeGeeX4 | 9B | Self-hosted multilingual coder |
| mistral-small-24b | Mistral Small | 24B nominal | Local instruct alias; exact checkpoint unresolved |
| internlm3-8b-instruct | InternLM3 | 8B | Self-hosted instruct line |
| kimi-dev-72b | Kimi-Dev | 72.7B public total | Multi-shard schedule on the H20 pool |
| glm-4.5-air | GLM-4.5 Air | 106B / 12B active | Tensor-parallel staged serving |

**Mistral checkpoint provenance.** The reported panel runs record `mistral-small-24b` at a local endpoint, without a contemporaneous checkpoint identifier or snapshot hash. A later SWE-Lite launcher maps that alias to a directory named `Mistral-Small-24B-Instruct-2501`, corresponding to Mistral Small 3 rather than Small 3.1 (Mistral AI, 2025a;b). That later mapping does not establish which checkpoint served the earlier panel. We therefore retain the recorded alias and nominal 24B scale; the exact checkpoint for this line remains unresolved.

### B.3 Benchmarks, profiles, and seeds

Table 13: Benchmark and profile details. Split entries are tune/holdout/canary/final task counts per seed. Anchor and Horizon share split geometry, but Horizon changes both generations and proposal breadth.

| Family | Usable tasks | Task form | Hidden evidence | Stress axis |
|---|---|---|---|---|
| MBPP+ | 378 | Function synthesis | EvalPlus hidden tests | Canonical public/hidden MBPP split |
| HumanEval+ | 164 | Function synthesis | EvalPlus hidden tests | Harder HumanEval anchor |
| MBPP Pro | 353 | Function synthesis | Held-out assertions | Harder MBPP-style partition |
| HumanEval Pro | 149 | Function synthesis | Held-out assertions | Harder HumanEval-style partition |
| ClassEval | 100 | Class synthesis | Held-out test methods | State and multi-method interaction |
| APPS | 64 | stdin/stdout synthesis | Held-out I/O cases | Execution-heavy program synthesis |
| **Profile** | **Split** | **Search** | **Compared variants** | **Intended role** |
| Anchor | 6/4/4/6 | 2 generations, 3 proposals | Seven variants | Low-depth reference |
| Horizon | 6/4/4/6 | 4 generations, 5 proposals | Four sequential variants | Deeper and wider search |
| Confirmatory | 10/8/8/10 | 2 generations, 3 proposals | Seven variants | Larger evidence profile |

The benchmark sources are HumanEval and MBPP (Chen et al., 2021; Austin et al., 2021), EvalPlus (Liu et al., 2023), the Pro self-invoking extensions (Yu et al., 2025), ClassEval (Du et al., 2023), and APPS (Hendrycks et al., 2021). The "usable tasks" column gives this study's retained pool, not necessarily each benchmark's full published size. All six families use seeds 11, 29, 47, 61, and 83. Variants within a model–benchmark–profile–seed cell share the same split manifest. MBPP+ uses a 3-second execution timeout; HumanEval+ and the Pro families use 5 seconds; ClassEval and APPS use 8 seconds. Hosted and self-hosted lines use a common OpenAI-compatible interface. Self-hosted models run through vLLM (Kwon et al., 2023). Decoding is deterministic, and the agent changes prompts/workflows only; no model weight, benchmark harness, or hidden test is modified.

# C Evidence Populations and Retry-Aware Contrasts

## C.1 Canonical population and denominator scope

The frozen six-family snapshot contains 24 model lines × six benchmark families × three profiles, with 432/432 result files. Its source hash is `6f45bd37de5467fc25a8b5faa974328f5b1032fd570ad0e69731c9d34d2e906a`. The core counts are 340/959, 918/1,924, and 414/851. The Fresh-LiveCodeBench extension contributes another 57/164, 6/9, and 3/4 and is kept separate because it has different coverage; pooling it with the core would instead yield 397/1,123, 924/1,933, and 417/855. The matched-Horizon endpoint tally within the core is 26/144 wins, 46 losses, and 72 ties.

Table 5 in Section 5.1 gives the retry-aware contrasts.

## C.2 First admissions and repetition

The first-admission counts are 145/577 (25.1%), 169/667 (25.3%), and 177/527 (33.6%) for Anchor, Horizon, and Confirmatory. These primary decision-level rates remain sizable, but they do not show a larger Horizon effect. The retry-inclusive difference instead reflects how often the realized process revisits a pair.

Table 14: Exact incumbent–candidate repetitions among later admitted events. The late Horizon process is dominated by pairs already evaluated in the same run.

| Generation | Repeated exact pairs / admitted events | Rate |
|---|---|---|
| Anchor generation 2 | 145/382 | 38.0% |
| Horizon generation 2 | 169/423 | 40.0% |
| Horizon generation 3 | 232/417 | 55.6% |
| Horizon generation 4 | 408/417 | 97.8% |
| Confirmatory generation 2 | 177/324 | 54.6% |

## C.3 Estimands and uncertainty

The raw, first-exact-pair, and first-admission estimands and the model-cluster bootstrap are defined in Section 3.3; the definitions are shared by the core and its separately reported sensitivities.

## C.4 Retry-inclusive heterogeneity across model lines

Figure 2 reports the raw operational rejection rate for every model–benchmark line. The matrix is deliberately interpreted as gate workload, not as 24 independent estimates of distinct-decision failure. Its value is descriptive: disagreement is not confined to one provider, model size, benchmark family, or serving path.

Table 15: Matched-Horizon line counts. Endpoint wins are uncommon, while rollback and conservative no-update behavior are widespread. A win count does not test significance or imply that ties are losses.

| Benchmark | SEED-Full > No Rollback | Any rollback | SEED-Tight no update |
|---|---|---|---|
| HumanEval+ | 4/24 | 21/24 | 22/24 |
| MBPP+ | 8/24 | 22/24 | 23/24 |
| HumanEval Pro | 4/24 | 21/24 | 23/24 |
| MBPP Pro | 3/24 | 22/24 | 24/24 |
| ClassEval | 5/24 | 24/24 | 20/24 |
| APPS | 2/24 | 16/24 | 24/24 |
| All lines | 26/144 | 126/144 | 136/144 |

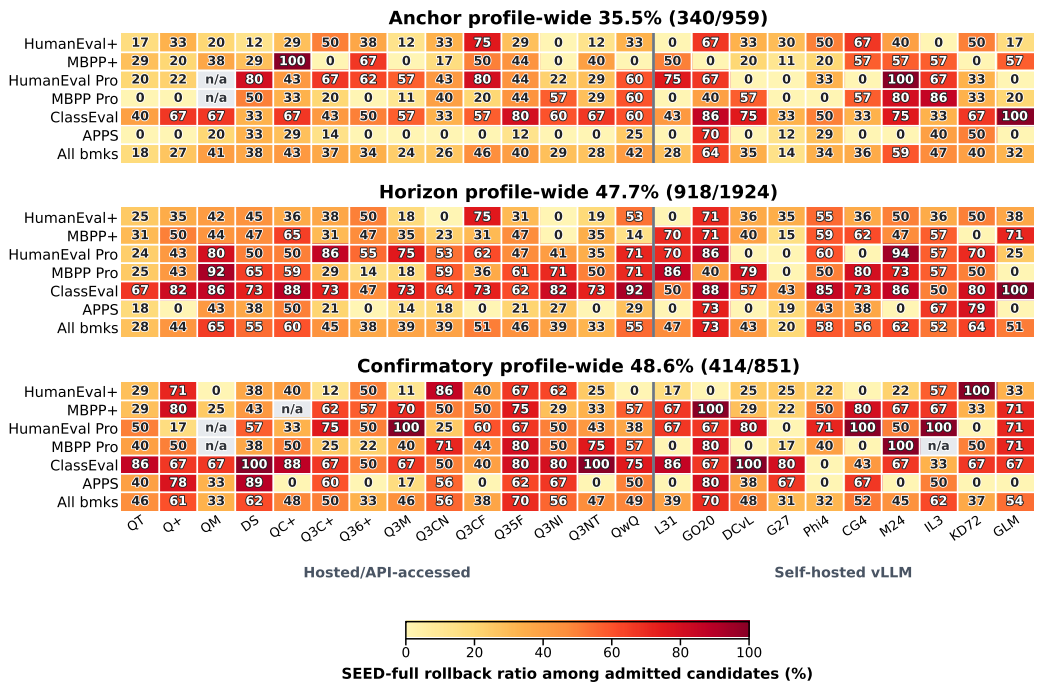


Figure 2: **Retry-inclusive canary disagreement across the 24-line core panel.** Each cell is the fraction of admitted SEED-FULL events rolled back on the canary. Profile-wide counts use the six-family core population. Late-search retries contribute to these rates, so they quantify operational burden rather than distinct-decision non-transfer.

## D CONDITIONAL TASK-REPARTITION DIAGNOSTIC

The self-hosted final-evidence subset stores all task outcomes required to reconstruct the realized canary: 1,375/1,375 retained events have complete coverage and zero invariant exclusions. We keep the realized candidate, incumbent, admission decision, and future trajectory fixed, then redraw conditional partitions from the retained outcomes. This estimates split disagreement conditional on the historical path; it is not a trajectory-level permutation null.

Table 16: Complete retained-outcome diagnostic. The all-admission rates are 84/317 versus 105/410. Cell support comprises 235 common-admission cells, 37 Horizon-only cells (40 events), and no Anchor-only cell.

| Population | Quantity | H–A pp [95% interval] |
|---|---|---|
| All admissions | Observed disagreement | $-0.89$ $[-3.70, +1.77]$ |
| All admissions | Conditional task repartition | $-0.39$ $[-2.01, +0.90]$ |
| All admissions | Difference in differences | $-0.50$ $[-1.86, +1.07]$ |
| Common cells | Observed disagreement | $-1.63$ $[-4.23, +1.76]$ |
| Common cells | Conditional task repartition | $-0.84$ $[-2.19, +0.80]$ |
| Common cells | Difference in differences | $-0.80$ $[-2.19, +1.06]$ |

The mixed-stack attempt is not used for inference: only 1,497/3,530 events (42.4%) are valid, 1,945 are missing, 88 are excluded, and 91.85% of valid events are self-hosted. Foregrounding that mixture would make missingness and stack composition load-bearing.

# E TRAJECTORY SENSITIVITY WITH FAILED-PAIR BLACKLISTING

Section 5.3 and Table 6 report the failed-pair-blacklist sensitivity and missing-outcome identification ranges. This replay changes the search path, unlike first-exact filtering and the conditional task-repartition diagnostic.

**Cell and seed-run coverage.** The replay comprises $10 \times 6 \times 3 = 180$ self-hosted model–benchmark-family–profile cells, each with five source seeds, for $180 \times 5 = 900$ planned seed-runs. A cell is counted as fully completed only when all five seed-runs finish. Thus 154/180 fully completed cells contribute $154 \times 5 = 770$ seed-runs. The remaining 26 cells are partial: 22 complete four seed-runs and four complete three, contributing $22 \times 4 + 4 \times 3 = 100$. Hence $770 + 100 = 870$ seed-runs are complete (870/900), with 30 unfinished (Table 17). All 60 Anchor cells are fully completed. The 26 partial cells comprise 21 Horizon and five Confirmatory cells.

Table 17: Frozen blacklist-replay completion accounting. A cell is a model–benchmark-family–profile combination with five planned seed-runs. Completed runs from partial cells contribute to seed-run coverage; they do not turn those cells into fully completed cells.

| Cell completion | Cells | Completed seed-runs | Unfinished seed-runs |
|---|---|---|---|
| 5 of 5 seed-runs | 154 | 770 | 0 |
| 4 of 5 seed-runs | 22 | 88 | 22 |
| 3 of 5 seed-runs | 4 | 12 | 8 |
| Total | 180 | 870 | 30 |

**Paired-contrast coverage.** The Anchor–Horizon target consists of 60 model–family lines with five seeds each, or 300 possible run pairs. All Anchor seed-runs are complete; 275 have a completed Horizon partner and 25 do not. Matching is at the seed-run level, so a completed seed-run can contribute even when another seed in its cell is unfinished. The 25 unavailable Horizon branches are therefore unmatched seed-runs, not the 26 partial cells or the 30 unfinished seed-runs across all profiles. The observed-pair contrast retains 78/315 versus 136/513 rollbacks/admissions and $+1.75$ percentage points with interval $[-1.27, +6.14]$. The missing-outcome bounds instead refer to the full 300-pair target.

**Frozen-response boundary and missingness.** Each of the 30 unfinished seed-runs stopped because a required frozen response file was missing. These outcomes remain unobserved; no newly generated responses are substituted to complete this replay. Additional model calls would create new evidence rather than recover the missing frozen records. A file-availability cause does not establish outcome-independent missingness: the observed Anchor histories associated with the 25 unmatched Horizon branches contain 35 rollbacks among 48 admissions (72.92%). We therefore retain the observed-pair contrast and the full-target missing-outcome bounds as distinct summaries, without claiming that the observed subset identifies a complete-replay or one-factor causal effect.

# F LOCAL UNTOUCHED-FINAL UTILITY

Table 7 in Section 5.4 gives the two-unit numerical summary. We distinguish pass-or-gap protection, direct-pass protection, and recall. Gap-only cases are useful for evaluating the declared gate but should not be described as capability loss. Direct-pass events count local task losses on the small untouched final split, not downstream deployment outcomes.

First-exact recall is 271/799 (33.9%) for pass-or-gap regressions and 159/478 (33.3%) for direct-pass regressions. Among the 159 direct-pass decisions, 110, 35, 11, and three lose exactly one, two, three, and four final tasks, respectively; 49/159 lose at least two and 14/159 lose at least three. These counts describe local severity but do not justify selecting a new threshold on the final split. Any replacement policy requires a separate policy-selection surface followed by new untouched evaluation.

### F.1 Operational diagnostics by benchmark family

Figure 3 separates four quantities that need not move together: accepted updates, retry-inclusive rollback events, token use, and seed-level endpoint dispersion. Horizon's deeper and wider search increases accepted-update and rollback exposure, while Confirmatory's larger task budget raises token use and often lowers seed dispersion. These plots describe workload and variability; they are not evidence that repeated rollback events are independent decisions.

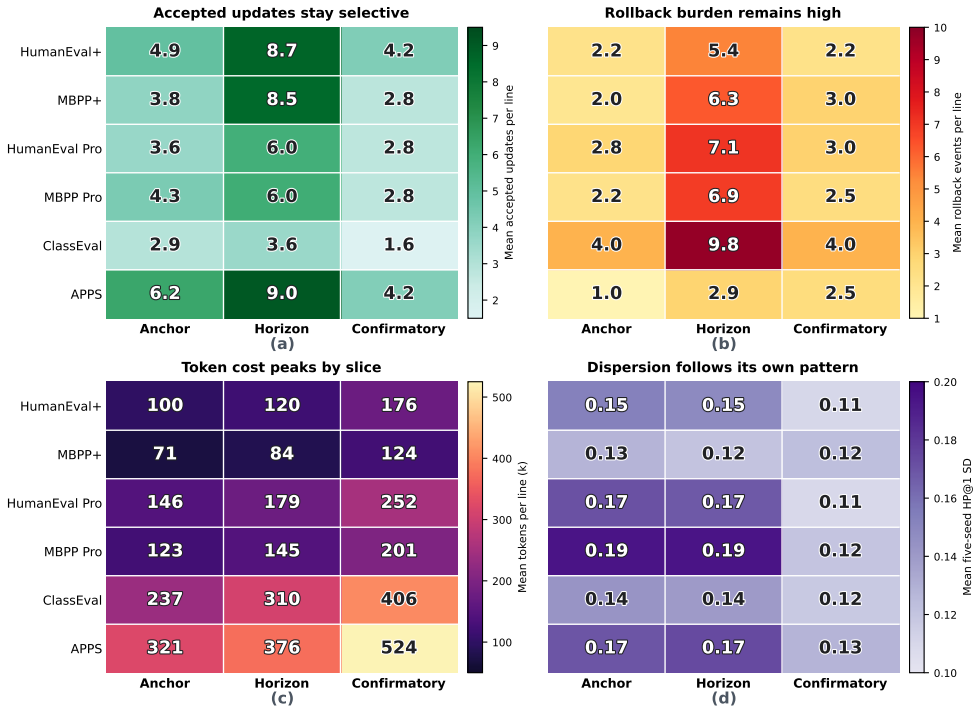


Figure 3: **Operational and endpoint diagnostics across benchmark families.** Accepted updates and rollback events are retry-inclusive means per model line. Token counts are logged usage, and dispersion is the standard deviation of five-seed final `HP@1` (NIST/SEMATECH, n.d.). Together they separate operational volume, resource use, and endpoint variability.

### F.2 One realized six-benchmark trace

Table 18 gives one qualitative trace for the dated Qwen3.6-Plus model, seed 11, under Anchor. It is an illustration, not evidence that this line is representative. The six rows contain accepted upgrades, no-ops, and rollbacks. In MBPP+, the same exact candidate is reconsidered and rolled back twice, making the event/decision distinction concrete. In APPS, two candidates pass confirmation and the second becomes the final incumbent.

## G Fixed-Budget Allocation Sensitivity

Tables 8 and 9 in Section 5.5 give the operational and endpoint results. The fixed five-task experiment covers 57/60 possible self-hosted model–benchmark-family lines across ten models and five seeds. The two predeclared hosted lines are excluded because their four-allocation caches are incomplete.

The ten-model endpoint intervals cross zero under every allocation comparison. The evidence supports neither an endpoint optimum nor equivalence among allocations. It instead shows that task allocation changes the number of provisional admissions, rollback checks, deployments, protective events, and realized calls/tokens.

Table 18: A single realized trace. T/H/C denotes tune, holdout, and canary pass rates; SR is `self_review_guarded` and PTR is `public_test_repair`. The example illustrates state transitions and exact-pair retries; it is not used to estimate panel-wide effects.

| Benchmark | Generation 1 | Generation 2 | Deployment reason | Final agent | Final HP@1 |
|---|---|---|---|---|---|
| HumanEval+ | `balanced_seed`; T/H/C 1.00/1.00/1.00; accept | No admissible candidate | Canary non-worse, then no-op | `balanced_seed` | 1.0000 |
| MBPP+ | SR; 1.00/0.50/0.75; rollback | Same exact pair; 1.00/0.50/0.75; rollback | Canary pass $-0.25$, gap $+0.25$ | `base_direct` | 0.8333 |
| HumanEval Pro | No admissible candidate | – | Holdout no-op | `base_direct` | 0.8333 |
| MBPP Pro | No admissible candidate | – | Holdout no-op | `base_direct` | 0.3333 |
| ClassEval | `balanced_seed`; 0.50/1.00/0.75; rollback | Same exact pair; 0.50/1.00/0.75; rollback | Canary pass tie, gap $+0.25$ | `base_direct` | 0.3333 |
| APPS | PTR; 0.83/1.00/1.00; accept | `balanced_seed`; 0.83/1.00/1.00; accept | Canary non-worse twice | `balanced_seed` | 1.0000 |

# H Freshness and Repository-Scale Checks

## H.1 Fresh LiveCodeBench

The matched Anchor/Horizon extension uses four hosted Qwen-family lines. Raw Horizon–Anchor disagreement is $+25.16$ points ($[9.29, 38.37]$), with exact model sign $p = 0.125$. First-exact disagreement is $+8.47$ points ($[-8.97, 21.67]$), with $p = 1$. Horizon contains 4/21 raw and 1/6 first-exact protective rollback decisions. Four lines do not provide a decisive family- level test, and public-benchmark exclusion does not detect memorization.

## H.2 SWE-bench Lite

The 23-line `sympy/sympy` pilot lies near an outcome floor. In that regime, SEED-Tight often records no-op rather than shipping an update without hidden-evidence lift. This is evidence that the trace represents abstention; it is not an estimate of repository-scale rollback precision or multi-file software-engineering benefit.

## H.3 Split size

Confirmatory uses 8/8 holdout/canary tasks and retains a 177/527 (33.59%) first-admission disagreement rate. It also changes tune/final sizes and the overall evidence profile, so it is not a matched size-only comparison. A separate exact-$k = 8$ reproduction did not preserve the intended common target population and is omitted. Split-size sensitivity remains unresolved.

# I Compute and Cost Scope

Calls and tokens are stable logged quantities; billed dollars are not, because provider aliases, public price cards, caching, and dated snapshots can change. Table 19 therefore reports the 14-line hosted-access slice in calls and total tokens. Self-hosted vLLM lines are excluded from this table.

On directly comparable hosted Anchor+Confirmatory slices, SEED-Full totals about 104,133 calls and 231.6M tokens, compared with 8,765 calls and 18.2M tokens for Reflection-Only. This large resource difference prevents treating the protocol as a cost-free safeguard. Fixed-budget results further show that allocation can change realized compute through earlier rejection, so cost and policy behavior must be reported together.

# J Reproducibility Record

The anonymous review artifact can be accessed at https://anonymous.4open.science/r/seed-review-artifact-D056.

Table 19: Profile-level hosted usage. Horizon omits Frozen, REFLECTION-ONLY, and Unconstrained by design.

| Profile | Variant | Mean calls/line | Mean tokens/line (k) | Total calls | Total tokens (M) |
|---|---|---|---|---|---|
| Anchor | Frozen | 6.0 | 10.8 | 2,520 | 4.5 |
| Anchor | REFLECTION-ONLY | 8.1 | 17.4 | 3,396 | 7.3 |
| Anchor | Unconstrained | 103.6 | 236.6 | 43,500 | 99.4 |
| Anchor | SINGLE EVALUATOR | 100.4 | 230.4 | 42,178 | 96.8 |
| Anchor | NO ROLLBACK | 95.8 | 219.8 | 40,243 | 92.3 |
| Anchor | SEED-FULL | 90.8 | 205.9 | 38,116 | 86.5 |
| Anchor | SEED-TIGHT | 69.5 | 150.1 | 29,204 | 63.1 |
| Horizon | SINGLE EVALUATOR | 114.5 | 257.4 | 48,073 | 108.1 |
| Horizon | NO ROLLBACK | 111.2 | 251.9 | 46,688 | 105.8 |
| Horizon | SEED-FULL | 109.6 | 246.6 | 46,025 | 103.6 |
| Horizon | SEED-TIGHT | 86.5 | 191.1 | 36,331 | 80.3 |
| Confirmatory | Frozen | 10.0 | 17.5 | 4,200 | 7.4 |
| Confirmatory | REFLECTION-ONLY | 12.8 | 25.8 | 5,369 | 10.8 |
| Confirmatory | Unconstrained | 184.8 | 424.9 | 77,623 | 178.5 |
| Confirmatory | SINGLE EVALUATOR | 182.0 | 410.1 | 76,445 | 172.3 |
| Confirmatory | NO ROLLBACK | 169.0 | 379.5 | 70,995 | 159.4 |
| Confirmatory | SEED-FULL | 157.2 | 345.6 | 66,017 | 145.1 |
| Confirmatory | SEED-TIGHT | 127.2 | 271.7 | 53,412 | 114.1 |

The anonymized artifact contains the harness, benchmark adapters, mutation library, split manifests, cached model responses where policy permits, event rows, aggregation scripts, and figure/table exporters. A reproducible run requires the exact model alias or self-hosted checkpoint, serving stack, task snapshot, split manifest, seed, timeout, decoding configuration, proposal budget, admission rule, and artifact version. For the historical Mistral line, the recorded alias does not determine an exact checkpoint (Appendix B.2); its recorded results should not be interpreted as a fully pinned model reproduction.

For each deployment decision, event rows record the incumbent and candidate, tune/holdout/canary outcomes, admission and rollback decisions, generation, final reporting outcomes, call counts, and token counts. Task identifiers are stored in the seed-level split manifest, including the fixed canary list; they are not duplicated in every event row. The trace combines these linked records, and the retry analysis additionally records an exact-pair key within each run. These fields support three separate exports: retry-inclusive events, first admissions, and first exact pairs. Reproducing a headline requires declaring which export is used.

The canonical six-family population, final-evidence utility subset, conditional repartition subset, blacklist replay, fixed-budget replication, Fresh-LiveCodeBench extension, and SWE-bench pilot have different coverage. They are therefore stored and reported as separate populations rather than joined into one headline table. Blacklist coverage distinguishes fully completed cells, completed seed-runs, and matched Anchor/Horizon pairs; missing frozen responses remain unobserved (Appendix E).

# K DEPLOYMENT TRACE SCHEMA

The minimum candidate-level trace is in Section 3.4, Table 3. The reporting requirements below use that same schema.

## K.1 MINIMUM AGGREGATE REPORT

A paper or deployment report using this schema should provide four linked summaries. First, it should state the task populations and denominator filters for admissions, canary checks, and final-evidence coverage. Second, it should report retry-inclusive events, first admissions, and first exact pairs separately. Third, any protective interpretation should include direct-pass and pass-or-gap precision

together with recall. Fourth, endpoint effects should be reported under matched budget and resource conditions, with null intervals retained.

These requirements prevent several common ambiguities. A high rollback count can arise from many distinct failed decisions or repeated exposure of the same pair. A low endpoint delta can coexist with frequent local protection, but it can also reflect unnecessary rollback. A clean canary split can be disjoint in the current decision while still influencing later search through the surviving incumbent. The trace makes each distinction explicit without asserting that one gate or budget is universally preferred.

Table 20: Evidence-scope summary. The table states the strongest conclusion that each current population supports.

| Claim or question | Current evidence | Interpretation |
|---|---|---|
| Does canary confirmation change decisions? | First-admission and first-exact counts over the 24-line core | Yes, locally and frequently enough to report the trace |
| Does deeper/wider search increase distinct non-transfer? | First-exact and conditional-repartition intervals cross zero | Not established; raw increase is retry-inclusive burden |
| Does rollback prevent every local regression? | First-exact recall 33.9% P/G and 33.3% direct pass | No; the gate is partial |
| Is one fixed budget split endpoint-optimal? | All final-`HP@1` intervals cross zero | Not established |
| Is the core contamination-free? | Public-family exclusion and four-line fresh sensitivity | Unresolved |
| Does the protocol transfer to repository-scale coding? | Floor-saturated SWE-bench Lite pilot | Precision unresolved; no-op behavior only |
| Does it transfer beyond coding? | No non-coding experiment | Unresolved |

## L Validity Boundaries

**Construct validity.** `HP@1`, visible–hidden gap, rollback, no-op, and local final regressions measure evaluator transfer and deployment-control behavior. They do not directly measure maintainability, latency, exploitability, privacy, calibration, human acceptance, or downstream user outcomes. A canary failure is disagreement with a sampled hidden slice, not evidence about those broader outcomes.

**Analysis unit.** History-conditioned search can repeat exact pairs. Retry-inclusive events are appropriate for operational workload but overstate the number of distinct decisions. First-exact filtering reduces this duplication without rerunning the trajectory; it does not create a globally unique-decision policy or remove all history dependence.

**Internal validity.** Within-cell split sharing, deterministic decoding, fixed seeds, and explicit task roles reduce confounding. Anchor and Horizon nevertheless change both generation depth and proposal breadth. The canary list is reused within each run, and rollback changes future incumbents; task disjointness therefore does not imply independence across generations. Conditional repartition fixes the realized path, while blacklist replay changes the path but has incomplete coverage because frozen responses are missing (Appendix E). Neither alone identifies a pure effect of adaptive reuse.

**Statistical conclusion validity.** Per-seed hidden splits are small and discrete. We cluster uncertainty at the model level and retain null intervals, but the number of independent model families is smaller than the number of aliases. Fresh-LiveCodeBench contains only four matched lines. The fixed-budget panel covers ten self-hosted models and 57/60 lines; it is informative but does not establish equivalence.

**Contamination and freshness.** Excluding HumanEval+/MBPP+ or adding public fresh tasks does not detect training contamination. Uneven memorization could either mask or create split

disagreement. The core within-benchmark findings must therefore not be interpreted as contamination-free estimates.

**External validity.** The study covers bounded prompt/workflow search for coding tasks. The mutation library is hardcoded, the repository-level pilot is floor-saturated, and no non-coding agent loop is evaluated. Quantitative rates should not be exported to open-ended self-modification, web navigation, multimodal tool use, or general software engineering without new evidence.

**Hosted-system drift.** Provider aliases, routing, moderation layers, caching, and backend updates may change after data collection. Dated aliases are retained when available, and self-hosted checkpoints improve reproducibility where recorded, but the historical Mistral line lacks an exact checkpoint identifier (Appendix B.2). No hosted result is permanently invariant.

**Responsible interpretation.** The final-split operating curve is descriptive. Using it to tune a new gate would consume the evidence reserved for evaluation. Any deployment policy chosen from these results requires prospective validation on a new untouched surface. Current evidence therefore supports prospective validation and bounded conclusions rather than a universal deployment recommendation.

# M Extended Related Work and Scope Distinctions

The references below distinguish the present measurement problem from adjacent research directions. They provide background and comparisons, not additional experiments, baselines, safety guarantees, or evidence for the numerical results reported in this paper.

## M.1 Base models, training, and the object being updated

**Model development versus harness updates.** The Transformer architecture (Vaswani et al., 2017), generative pre-training (Radford et al., 2018), bidirectional language-model pre-training (Devlin et al., 2019), and the text-to-text transfer framework (Raffel et al., 2020) concern the construction and adaptation of language models. Few-shot prompting (Brown et al., 2020) provides a separate route to changing task behavior without fitting weights on the task. The GPT-4 report and the LLM survey describe broader model capabilities and development practices (Achiam et al., 2023; Zhao et al., 2026). Here the unit being updated is instead a bounded prompt/workflow configuration around a fixed model line. These distinctions prevent interpreting a harness update as a new pre-trained model or a weight-space learning method.

**Model-family lineage is not checkpoint identification.** The original LLaMA paper (Touvron et al., 2023), the CodeGeeX paper (Zheng et al., 2023), and the InternLM2 report (Cai et al., 2024) provide historical background for model families represented by later releases in the panel. They are not substitutes for the Llama 3, CodeGeeX4, and InternLM3 sources associated with the actual roster in Appendix B. A family paper, a model card, an API alias, and a pinned checkpoint serve different provenance roles.

**Learning from preferences.** Reinforcement learning optimizes behavior using interaction and reward (Sutton & Barto, 1998); AlphaGo combines learned networks and search in a domain with explicit game outcomes (Silver et al., 2016). Learning from human preferences (Christiano et al., 2017), language-model fine-tuning from preferences (Ziegler et al., 2019), and human-feedback summarization (Stiennon et al., 2020) illustrate distinct training settings. Instruction following with human feedback (Ouyang et al., 2022), alignment research with a general assistant (Askell et al., 2021), and helpful/harmless assistant training (Bai et al., 2022a) further separate task performance from the feedback used to select or train a system. Constitutional AI uses AI feedback in alignment (Bai et al., 2022b). SEED neither performs these training procedures nor substitutes its functional canary for their alignment objectives.

### M.2 Reasoning, feedback, tools, and agent-design search

**Within-task refinement versus cross-update evidence.** Self-Refine iterates generation, feedback, and refinement (Madaan et al., 2023); Reflexion uses verbal feedback in language-agent learning (Shinn et al., 2023). Tree of Thoughts explores reasoning branches (Yao et al., 2023a), while self-consistency aggregates sampled reasoning paths (Wang et al., 2023). Decomposed prompting makes complex tasks modular (Khot et al., 2022). These mechanisms motivate possible workflow components, but our static Reflection-Only comparator and six-library search are not reproductions of all these algorithms. We ask what evidence remains available after an update is selected, rather than assuming that refinement is reliably beneficial.

**Tools and multiple agents.** ReAct interleaves reasoning and actions (Yao et al., 2023b), and Toolformer studies learning tool use (Schick et al., 2023). Tool-learning and augmented-language-model surveys describe broader augmentation mechanisms (Qin et al., 2024; Mialon et al., 2023). CAMEL explores communicative agents (Li et al., 2023), and Generative Agents studies interactive simulations of human behavior (Park et al., 2023). Agent surveys and practitioner synthesis cover planning, memory, action, and their orchestration (Xi et al., 2025; Wang et al., 2024; Weng, 2023). The coding-loop measurements here do not establish transfer to these broader architectures, social simulations, or tool environments.

**What recent self-improvement work measures.** SICA, Darwin Gödel Machine, and Huxley-Gödel Machine investigate self-improving coding agents (Robeyns et al., 2025; Zhang et al., 2026; Wang et al., 2026); SEW concerns self-evolving code-generation workflows (Liu et al., 2025). Research-coding automation is also studied in the self-adapting-agent workshop paper (Gangireddi et al., 2026). RLEF and ReVeal train models to exploit execution feedback or verification (Gehring et al., 2025; Jin et al., 2026), unlike our fixed-weight comparison. SEA-Eval explicitly considers evaluation beyond isolated episodes (Jiang et al., 2026). VeRO and HarnessOpt-Bench provide infrastructure or benchmarks for harness optimization (Ursekar et al., 2026; Anonymous, 2026). These are substantive neighbors of our trace-based question, not evidence that our canary is new or that our six-module search reproduces open-ended evolution.

### M.3 Benchmark evidence, shift, and statistical uncertainty

**Benchmark design and contamination.** Behavioral testing with CheckList (Ribeiro et al., 2020), dynamic benchmark construction in Dynabench (Kiela et al., 2021), and the theory of dynamic benchmarks (Shirali et al., 2023) examine ways to move beyond a single static score. The benchmark lottery discusses how benchmark choices shape apparent progress (Dehghani et al., 2021). Prompt-sensitive evaluation is also examined by Inger et al. (2025). Such concerns are distinct from the specific within-run admission–canary disagreements studied here. LiveCodeBench (Jain et al., 2025), SWE-bench Goes Live! (Zhang et al., 2025), and SWE-Bench Pro (Deng et al., 2025) expand evaluation settings or collection procedures; they do not make a separate historical panel contamination-free. Contamination requires its own evidence (Magar & Schwartz, 2022; Deng et al., 2024).

**Capabilities outside the coding panel.** MMLU, BIG-bench, and HELM broaden language-model evaluation across tasks or measurement dimensions (Hendrycks et al., 2020; Srivastava et al., 2023; Liang et al., 2023). AgentBench evaluates interactive agent settings (Liu et al., 2024); WebArena and Mind2Web study web environments and web-agent tasks (Zhou et al., 2024; Deng et al., 2023). GAIA evaluates general-assistant tasks (Mialon et al., 2024). These benchmarks illustrate concrete directions for external validation. None is a non-coding experiment conducted in this paper, and none licenses exporting the reported local rollback rates to its domain.

**Distribution shift is not synonymous with adaptive reuse.** Out-of-distribution detection (Hendrycks & Gimpel, 2017), common-corruption robustness (Hendrycks & Dietterich, 2019), and NLP generalization under shift (Yang et al., 2023) pose different evaluation questions. Recollected test sets (Recht et al., 2019), underspecification (D'Amour et al., 2022), and WILDS (Koh et al., 2021) highlight additional limits of score transfer. Invariant risk minimization (Arjovsky et al., 2019) and domain-generalization evaluation (Gulrajani & Lopez-Paz, 2020) study cross-environment

learning; test-time training and Tent adapt models using test-time information (Sun et al., 2020; Wang et al., 2020). Our proposal-history restrictions do not implement those adaptation methods or control all forms of distribution shift.

**Continual learning and changing environments.** Catastrophic forgetting and continual learning concern preserving learned capabilities across training experiences (Kirkpatrick et al., 2017; Parisi et al., 2019). Learning from data streams addresses sequentially arriving data (Bifet et al., 2018). Delacroix's practitioner essay separately discusses the passage of time and changing social expectations around deployed systems (Delacroix, 2022). These sources motivate attention to time and state, but a rollback in our fixed-weight harness is not evidence of weight-space forgetting or a measure of changing social values.

**Predictive uncertainty versus uncertainty of an audit statistic.** Deep ensembles and dropout-based Bayesian approximation estimate forms of predictive uncertainty (Lakshminarayanan et al., 2017; Gal & Ghahramani, 2016). Model self-knowledge and unsupervised latent-knowledge extraction study other signals (Kadavath et al., 2022; Burns et al., 2023). Neither those signals nor the five-seed standard deviation (NIST/SEMATECH, n.d.) is identical to our model-cluster bootstrap interval. Bootstrap background (Efron, 1992) and cluster-resampling analysis (Field & Welsh, 2007) explain the resampling perspective, but do not make related provider aliases independent or validate an untested choice of clustering unit.

### M.4 Safety, lifecycle controls, and reproducibility

**Functional tests are not a comprehensive safety measure.** Research agendas for beneficial AI (Russell et al., 2015), risks of large-scale language models (Bender et al., 2021), and foundation-model opportunities and risks (Bommasani et al., 2021) address broader questions than executable correctness. Hallucination and mitigation surveys (Ji et al., 2023; Tonmoy et al., 2024) concern reliability dimensions not exhausted by a coding pass rate. Jailbreaking (Wei et al., 2023) and language-model red teaming (Perez et al., 2022; Ganguli et al., 2022) require adversarial evaluation that our functional canary does not provide. Misevolution explicitly examines risks of self-evolving agents (Shao et al., 2026); it motivates, but does not validate, a deployment control.

**Oversight and human outcomes.** Learned-optimization risks (Hubinger et al., 2019), scalable reward modeling (Leike et al., 2018), debate (Irving et al., 2018), and self-critiquing models assisting evaluators (Saunders et al., 2022) address distinct oversight mechanisms. User expectations of imperfect AI (Kocielnik et al., 2019) and sources of harm across the ML lifecycle (Suresh & Guttag, 2021) further explain why passing our final slice should not be equated with human acceptance or absence of downstream harm.

**Release engineering and audit records.** Technical debt (Sculley et al., 2015), production-readiness tests (Breck et al., 2017), practical ML engineering rules (Zinkevich, 2017), and automated data-quality verification (Schelter et al., 2018) motivate inspectable operational records. Production canarying (Warner & Davidovič, 2018), evaluation-driven agent operations (Xia et al., 2024), and governed capability evolution (Qin et al., 2026) are direct lifecycle-control precedents. Model cards and reproducibility guidance (Mitchell et al., 2019; Pineau et al., 2021) complement our separation of task, model, and trajectory provenance. For software-based evaluation, a testing framework such as pytest (Krekel et al., 2004) is a separately versioned dependency, not itself a benchmark definition or a guarantee of test adequacy. This general tooling citation does not assert a specific pytest version or an unreported experiment; the release manifest remains authoritative for usage.