LEARNING TO VERIFY BY DOING: A SELF-RENEWING POLICY–VERIFIER LOOP FOR SUSTAINED SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Verifier-guided self-training can stall as policy improvement erodes the selection advantage of existing evaluative guidance. We show that execution feedback can renew this guidance by teaching a verifier about action outcomes and applicability conditions. We introduce Knowing–Doing Co-Evolution (KDCE), a self-renewing policy–verifier loop that learns outcome prediction and candidate ranking from budgeted executions and controlled local action comparisons, then uses uncertainty-aware selection for weighted policy self-training. The updated policy supplies experience for the next round, closing the loop. Across 8 checkpoints from 3 model families and 11 code, mathematical-reasoning, and interactive evaluations, KDCE improves family- and domain-balanced standalone performance by 0.7 percentage points over prespecified domain-specific references after 8 rounds. Matched-evidence and directional interventions support both feedback paths. Gains are heterogeneous: improved verification does not always yield policy gains or compute savings.

1 INTRODUCTION

Self-improvement turns a model's own attempts and feedback into supervision for reasoning, code generation, and interactive decisions (Singh et al., 2023; Shinn et al., 2023; Yuan et al., 2024). In verifier-guided self-training, a policy generates candidates, a verifier selects promising ones, and the policy learns from those selections. This process relies on selected targets being more useful than typical policy outputs (Song et al., 2025). As the policy improves, that advantage can shrink, leaving additional sampling and training with diminishing returns.

The challenge is therefore not just to improve the policy, but to renew the judgments that guide it. We call the ability to judge actions and predict their consequences *knowing*, and the ability to act successfully *doing*. Building on joint generation–evaluation learning (Yuan et al., 2024; Zhang et al., 2024), we ask: *can a model's own actions supply knowledge that improves verification and renews guidance for further learning?*

Our observations suggest a way to renew this guidance (Section 2). Larger candidate pools can contain more correct solutions without a corresponding gain in selection accuracy. By contrast, execution feedback improves verification on unseen candidates when it preserves useful action–outcome relationships and applicability conditions. Execution can therefore do more than filter successful outputs: it can teach the verifier which actions work and under which conditions.

We therefore propose **Knowing–Doing Co-Evolution (KDCE)** (Section 3). Within an execution budget, each round executes proposed actions and selected local alternatives, updates outcome prediction and candidate ranking, and uses uncertainty-aware selection to construct policy training targets. Weighted self-training then updates the policy, which supplies experience for the next round. A conditional single-round analysis links improvement to target quality, verifier error, and the policy's ability to learn from those targets.

We evaluate KDCE on 8 checkpoints from 3 model families across 11 code, mathematical-reasoning, and interactive evaluations (Section 4). Standalone policy performance is measured separately from verifier-assisted inference. After 8 rounds, the family- and domain-balanced standalone gain

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.

over prespecified domain-specific references is 0.7 percentage points. Same-evidence comparisons, blocked and restored selection paths, and equal-cost exchanges of old- and new-policy experience test both directions of the loop. The results are heterogeneous: improved verification does not always benefit the policy, and the aggregate gains do not establish general compute savings.

Our contributions are:

- **Diagnosing feedback-limited improvement.** We separate candidate coverage, selection quality, and target absorption to identify when existing evaluative guidance becomes limiting.
- **Renewing verification through action.** KDCE turns budgeted execution evidence into updated judgments and policy learning targets in a reciprocal policy–verifier loop.
- **Testing both directions and their limits.** Resource-matched evaluations and directional interventions examine policy gains, experience quality, and failure conditions.

2 OBSERVATIONS

Building on the question introduced in Section 1, we examine learning from verifier-selected trajectories. Generation–verification gaps and sharpening motivate this mechanism (Song et al., 2025; Huang et al., 2025); when feedback quality lags behind policy improvement, additional sampling and training may yield diminishing returns. Where does this limitation arise, and can verifiable execution outcomes improve verification for subsequent policy learning?

First, training dynamics and sampling sweeps test whether shrinking selection advantages accompany diminishing gains; matched interventions isolate coverage, ranking, and policy learning. Next, same-trajectory feedback, counterfactual, and task-held-out comparisons test whether execution evidence teaches transferable verification and selective reuse reduces negative transfer. Finally, multi-round, budget-matched comparisons test whether verifier updates restore selection advantages and support policy gains. These hypotheses inform KDCE's consequence learning, experience reuse, and alternating updates.

We study mathematics, code, and tool interaction with objective offline audits. Diagnostic labels remain separate from routine training feedback; compute-matched and environment-call-matched comparisons are separate protocols. Fixed held-out tasks and dynamically remaining unsolved tasks distinguish within-task change from difficulty shifts (Appendix A).

2.1 EMPIRICAL ANALYSIS: FROM VERIFIER-GUIDED IMPROVEMENT TO VERIFIER LEARNING

2.1.1 DIMINISHING GAINS FROM VERIFIER-GUIDED SELF-TRAINING

At fixed candidate count N, D is pass@1, C is oracle pass@N (a pool contains a correct candidate), R is the probability of selecting a correct candidate conditional on availability, and K is best-of-N accuracy. For binary outcomes and exactly one selection,

$$K_t(N) = C_t(N)R_t(N), \qquad G_t(N) = K_t(N) - D_t. \tag{1}$$

Here G is our selected-versus-single-sample accuracy difference. We operationalize verification capability through outcome prediction and candidate ranking; online K also depends on the policy (Appendix A.3).

We track these quantities across models, domains, and training methods on fixed tasks, reporting unsolved subsets separately (Appendix B.1). The selection advantage G narrows alongside diminishing gains (Figure 1a). Motivated by solver–verifier dynamics (Sun et al., 2026), we test next-round gain prediction using $S = UG$, with update effectiveness U calibrated independently, against G, coverage, rollout count, and training loss (Figure 1b; Appendix B.2). The predictive comparison distinguishes UG from G alone. Can more sampling restore useful training data?

Holding checkpoints fixed, we vary N with nested pools. Oracle coverage rises while selection accuracy plateaus or declines (Figure 1c; Appendix B.1). Inserting verified solutions into matched unsolved-task pools tests whether correct candidates are overlooked; top-1 recognition is only 37.9%

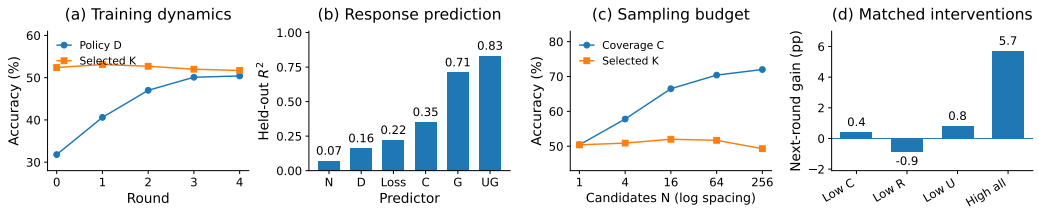

Figure 1: **Observation diagnostics.** Training dynamics, held-out response prediction, coverage–selection, and matched interventions characterize the diminishing-gain bottleneck (Appendix B).

(Appendix B.1). Same-pool oracle, external, self, and random selectors then isolate ranking differences, while matched retraining measures whether erroneous selections cause regression (Appendices B.1 and B.2).

We next hold initialization and update budgets fixed while varying candidate composition, selectors, and update protocols (Figure 1d; Appendix B.2). Coverage or ranking improvements alone remain limited by the other components, and identical positive G yields different gains under different update effectiveness. Together with the sampling decomposition (Appendix G.3), these interventions locate usable-feedback limitations. What evidence could teach the verifier judgments it currently misses?

2.1.2 Learning transferable verification from execution feedback

Actions determine which states a policy visits; verifiable consequences can supervise outcome prediction, as in CodeRL (Le et al., 2022). Holding trajectories and verifier initialization fixed, we compare replay, unchecked reflection, final outcomes, localized feedback, state–action–outcome records, and explanations checked against execution evidence. We compare pre- and post-update verifiers on identical unseen reference pools, measuring ΔK_{ref} separately from changing policy outputs (Appendix B.3). Grounded feedback improves ranking, with checked explanations providing the largest gain (Figure 2a).

Counterfactual action edits at fixed states, incorrect or irrelevant explanations paired with correct outcomes, and length-matched field masking test transferable relationships (Appendix B.3). Incorrect attribution weakens counterfactual generalization. Information proxies, prediction losses, deduplicated variants, and unseen combinations test learning beyond confidence and memorization (Appendix B.3). Together, they probe error causes, action effects, alternatives, and applicability conditions.

We then withhold all target-task training and development feedback to test transfer from related tasks (Appendix B.4). We examine whether transfer aligns with shared primitives beyond textual similarity (Appendix B.4). A separate experiment permits limited target feedback and compares task-only, pooled, retrieved, selective, and oracle reuse (Figure 2b; Appendix B.4). Unrelated and conflicting experience tests the effect of selection on negative transfer (Appendix B.4). Local verification gains leave open whether subsequent policy updates repeatedly benefit (Appendix B.4).

2.1.3 Translating verifier improvements into further policy gains

STV already links trained verification to generator improvement (Wu & Raghunathan, 2026). We examine eight-round execution-feedback loops using self-labeling, increased sampling, same-task feedback, global pooling, selective updates, and oracle updates (Appendix B.5). Self-labeling plateaus around round four, while selective updates support further gains (Figure 2c). Intermediate measurements distinguish online joint performance, fixed-reference verification, and old/new verifier replacement on identical current-round pools. Frozen and replacement branches test whether changed selection improves the subsequent policy beyond scheduled temporal ordering (Appendix D.1).

We vary feedback proportion, accuracy, and delay to assess how reliable feedback and relevant reuse affect late-stage stagnation and direct target-feedback demand (Figure 2d; Appendices B.5 and C.3). Incorrect supervision, forgetting, and information-source ablations test failure conditions and inter-

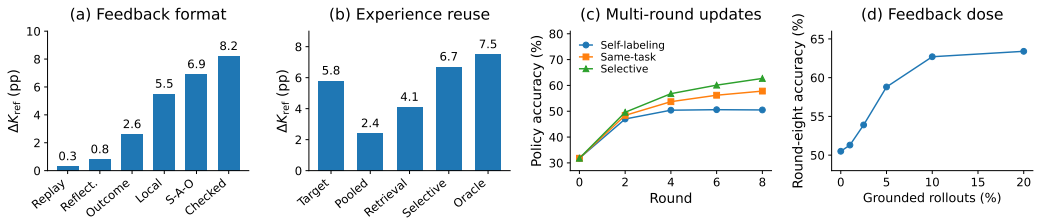

Figure 2: **Feedback diagnostics.** Grounded feedback, selective reuse, multi-round updates, and dose response characterize loop renewal and transfer risk (Appendices B.3–B.5).

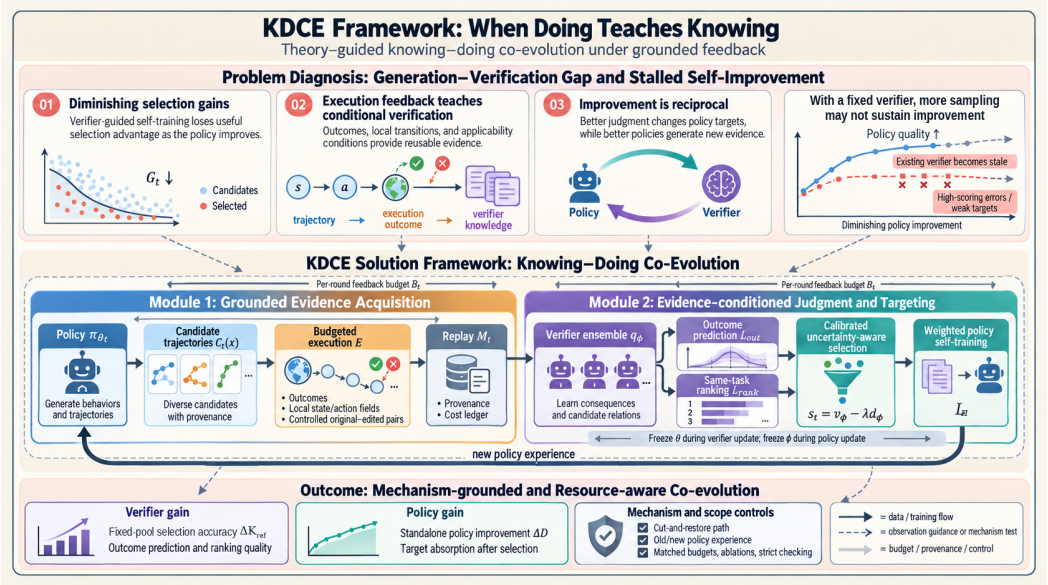

Figure 3: **KDCE framework.** Observed selection limits motivate budgeted execution, evidence-conditioned verification, uncertainty-aware target selection, and weighted policy self-training in a reciprocal policy–verifier loop.

actions (Appendix E.2). Large-budget pass@k, unseen combinations and states, and audits beyond finite base candidate pools test accessible capability beyond simple reranking (Appendix B.4); any gains establish finite-budget generalization, not support-set expansion.

Under a common protocol, $G_{t+1} - G_t = \Delta K_t - \Delta D_t$ tracks selection advantage; replenishment requires fixed-pool and same-round replacement evidence. Together, these tests connect feedback-limited stagnation, transferable verification, and further policy gains. Section 3 develops KDCE's exploration, consequence learning, explanation checking, experience reuse, and reliable alternating updates from these motivations.

3 METHOD

Building on the observations in Section 2.1, we propose *Knowing–Doing Co-Evolution* (KDCE). The policy generates trajectories, the verifier predicts consequences and ranks candidates, and budgeted execution supplies evidence before the next policy update. Each round follows the same causal chain: generate, acquire evidence, update the verifier, reselect targets, update the policy, and store experience. Thus, when behavior absorbs existing guidance faster than that guidance grows, doing supplies the evidence needed to improve knowing.

Figure 3 instantiates this observation-to-method mapping: grounded execution renews conditional judgment, calibrated uncertainty guides target selection, and weighted self-training closes the reciprocal policy–verifier loop.

3.1 Recursive Self-Improvement Framework

KDCE keeps the generate–filter–train pattern of iterative self-training (Singh et al., 2023), but addresses the generation–verification gap by letting execution evidence renew the filter. This connects precedents for joint generation and evaluation, including Self-Rewarding Language Models and V-STaR (Yuan et al., 2024; Hosseini et al., 2024), to the observation that renewed judgments must affect later policy targets; matched-supervision controls are in Appendix A.2.

Let $x \sim \mathcal{T}$ be a task with observable context and $\tau \sim \pi_{\theta_t}(\cdot \mid x)$ a round-t trajectory. Execution gives $o \sim \mathcal{E}(\cdot \mid x, \tau)$, scored by fixed utility $u(o) \in [0, 1]$; the verifier models consequences with q_ϕ and ranks by

$$v_\phi(x, \tau) = \mathbb{E}_{o \sim q_\phi(\cdot \mid x, \tau)}[u(o)]. \tag{2}$$

For binary tasks this is the predicted success probability; exposed tests, states, or error locations may add targets. The policy and verifier use separate adapters on a shared frozen backbone, while replay $\mathcal{M}_t$ stores executed trajectories, outcomes, and provenance (Appendix F).

The updates are separated as

$$(\theta_t, \phi_t) \xrightarrow{\text{execution evidence}} (\theta_t, \phi_{t+1}) \xrightarrow{\text{selected training examples}} (\theta_{t+1}, \phi_{t+1}). \tag{3}$$

Freezing the policy during verifier training and the verifier during policy training fixes the within-round selection rule. Only task inputs, generated trajectories, and outcomes queried within budget B_t are admissible; initialization, edits, calibration, and extra checks are charged, and training, calibration, and audit records are task-separated (Appendix A). The observation statistics C, R, K, G, U, S remain offline diagnostics, not labels or scheduling inputs.

3.2 Updating the Verifier from Execution Feedback

For each task, N trajectories form $\mathcal{C}_t(x)$. We query candidates by verifier disagreement per execution cost, reserve a random coverage share, and deduplicate records; this is an acquisition heuristic, not an exact information-gain estimator (Appendix F.2). Each record $e = (x, \tau, o)$ stores the environment, observed-field mask, and query ID. For a selected failure, paired executions of an original and locally edited trajectory from the same state identify the conditional edit effect; every new call is charged and new conditions test applicability (Appendix D.3).

Confirmed successes train the policy, while successes, failures, and local comparisons supervise the verifier, following execution-trained evaluation precedents such as CodeRL and V-STaR (Le et al., 2022; Hosseini et al., 2024). Updating the verifier before target selection is essential; same-evidence and failure-only controls isolate these two uses of feedback (Appendices D.1 and D.3). Let $\mathcal{E}_t^{\text{train}}$ mix current records with replay. Outcome prediction and same-task ranking use

$$\begin{aligned} \mathcal{L}_{\text{out}}(\phi) &= -\mathbb{E}_{(x,\tau,o) \sim \mathcal{E}_t^{\text{train}}} \log q_\phi(o \mid x, \tau), \\ \mathcal{L}_{\text{V}}(\phi) &= \mathcal{L}_{\text{out}}(\phi) - \alpha \mathbb{E}_{(x,\tau^+,\tau^-) \sim \mathcal{P}_t} \log \sigma\left(\frac{v_\phi(x, \tau^+) - v_\phi(x, \tau^-)}{T_{\text{r}}} \right), \end{aligned} \tag{4}$$

where $u(o^+) > u(o^-)$, $T_{\text{r}} > 0$, and $\alpha \geq 0$; ties supervise outcomes but not strict preference. The likelihood anchors consequences to observed evidence, while ranking supplies the distinctions needed for selection.

3.3 Verifier-Guided Policy Improvement

The updated verifier rescores the pool, including edits. Following the observed high-scoring-error failure mode, selection combines predicted utility and ensemble uncertainty:

$$s_t(x, \tau) = v_{\phi_{t+1}}(x, \tau) - \lambda d_{\phi_{t+1}}(x, \tau), \tag{5}$$

where d_ϕ is ensemble standard deviation and $\lambda \geq 0$; matched calibration tasks set the acceptance threshold. Confirmed successes take precedence, failures remain verifier evidence, and an unexecuted candidate is accepted only if it is task-wise highest-scoring and clears the threshold. Reserved checks target high-score, high-uncertainty candidates and override predictions; unexecuted selections retain predicted provenance and never become objective labels (Appendices F.3 and E).

Let $\mathcal{D}_t$ contain confirmed successes, accepted unexecuted candidates, successful edits, and successful replay. Policy distillation uses

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{(x,\tau,w)\sim\mathcal{D}_t}[w \log \pi_\theta(\tau \mid x)] + \eta \mathcal{R}_{\mathrm{KL}}(\theta, \theta_t). \tag{6}$$

Confirmed and unexecuted examples receive task-normalized raw weights 1 and $\omega \in (0, 1]$, respectively; η controls the token-level KL penalty and the verifier stays frozen. Empty task targets wait for later sampling, and the new policy supplies the next round of experience. Full round-wise pseudocode is given in Algorithm 1 in Appendix F.

The separation also gives a diagnostic for one-round improvement. Let $\rho_t(\tau \mid x)$ be the weighted target and $\widehat{A}_t$ its score advantage over π_{θ_t} under the updated verifier. If verifier error is at most $\varepsilon_t^{\mathrm{V}}$ on both distributions and the new policy is within average total variation ε_t^{π} of ρ_t, then

$$D_{t+1} - D_t \geq \widehat{A}_t - 2\varepsilon_t^{\mathrm{V}} - \varepsilon_t^{\pi}. \tag{7}$$

Appendix G defines the target and proves the bound. The two verifier-error terms cover the selected and original distributions, while ε_t^{π} measures target absorption; this is a one-round diagnostic, not a guarantee of indefinite improvement.

Matched interventions test both directions: updated judgment versus later policy learning, and old- versus new-policy experience versus verifier improvement (Appendices D.1 and D.3). Frequency, capacity, strict-checking, feedback, and forgetting controls delimit the claim (Appendices D.2 and E). The loop therefore closes as action, evidence, judgment, and next action.

4 Experiments

Evaluation scope. We evaluate KDCE under controlled, common-budget conditions. Five paired configurations define sensitivity ranges, not sampling confidence intervals. Independently published values retain their original protocols and are reported separately in Appendix C.1.

Setup. We evaluate 8 checkpoints from 3 model families on 11 code, mathematical-reasoning, and interactive evaluations, using standalone single-attempt success as the primary policy metric. Full model and task identities, data boundaries, baselines, inference controls, statistical units, aggregation rules, and resource accounting are specified in Appendix C.

4.1 Performance and efficiency

Motivated by the diminishing selection advantage identified in Section 2.1, Table 1 tests whether renewing verification is followed by stronger standalone policies. The prespecified reference is the online verifier for code and interactive settings and direct-outcome learning for mathematics. KDCE improves this family- and domain-balanced reference by 0.7 percentage points (pp), with a sensitivity range of [0.5, 0.9], and is positive in 16 of 24 model/domain settings; the positive-setting count uses unrounded paired differences. The effect is heterogeneous (-0.6 to 2.5 pp): mathematics is weakest, including a -0.6 pp deficit to direct-outcome learning for Qwen3.8-27B. This pattern supports a conditional, rather than universal, replenishment effect. Appendix C.2 separates standalone learning from selected@8 inference and retain the long-dependency condition in which online verification remains preferable.

However, renewed performance is not uniformly more efficient. Figure 4 shows that only 14 of 24 gain-per-compute ratios exceed one, with a family-balanced geometric mean of 0.99; 14 KDCE settings and 16 reference settings also miss their prespecified target within the shared cap. The complete budget grid and censored target analysis in Appendix C.3 therefore rule out a general compute-saving claim. The directional interventions that follow test the stronger mechanism suggested by the observations: whether renewed judgment changes subsequent learning and whether improved policies, in turn, supply more useful experience.

4.2 Additional Experimental Analyses

4.2.1 Does improved verification drive policy improvement?

The first directional test asks whether renewed judgments improve *future* learning beyond the effect of receiving execution feedback. From the same plateau checkpoint, Figure 5 yields 5.6 pp for

Table 1: Standalone success (%) after 8 rounds under common resource caps. Base receives no additional training, and each domain uses a separate checkpoint. Sensitivity ranges and results for all 8 models appear in Appendix C.

	Code		Recent mathematics		Interactive	
Method	LCB v6	LCB Pro	AIME26	HMMT26	SWE Pro v1	TB4.0
Qwen3.8-27B						
Base	86.0	30.0	86.0	70.0	46.0	30.0
Fixed verifier	87.2	31.0	86.8	71.2	47.2	31.3
Success filtering	87.8	31.5	87.4	72.1	48.5	32.6
Direct outcome	89.1	32.4	89.3	75.0	51.0	35.2
Online verifier	89.7	32.9	88.8	74.1	52.2	36.5
V-STaR-style (adapt.)	89.5	32.8	89.0	74.5	51.6	35.8
KDCE	90.1	33.2	88.6	73.9	54.2	38.5
Gemma 4 12B						
Base	70.0	13.0	68.0	50.0	24.0	15.0
Fixed verifier	71.9	14.1	69.4	51.7	25.5	16.5
Success filtering	72.9	14.7	70.3	52.8	27.1	17.9
Direct outcome	74.9	15.8	73.5	56.7	30.1	20.8
Online verifier	75.8	16.3	72.5	55.6	31.6	22.3
V-STaR-style (adapt.)	75.6	16.2	73.0	56.2	30.9	21.5
KDCE	77.3	17.2	73.4	56.6	33.6	24.1
Olmo 3 7B						
Base	36.0	1.5	40.0	23.0	6.0	2.0
Fixed verifier	39.1	2.7	42.0	25.2	7.7	3.6
Success filtering	40.6	3.3	43.3	26.6	9.5	5.2
Direct outcome	43.7	4.5	47.9	31.7	12.9	8.3
Online verifier	45.2	5.1	46.6	30.3	14.7	9.9
V-STaR-style (adapt.)	44.8	4.9	47.3	31.0	13.8	9.1
KDCE	47.8	6.1	47.6	31.3	14.1	9.4

KDCE, 1.0 pp for continued fixed-verifier training, and 3.4 pp for direct-outcome learning. Under identical execution records, the corresponding gains are 3.8, 3.2, and 3.6 pp for KDCE, online verification, and trace-to-policy repair. The remaining 0.6 pp increment over the online verifier isolates information use after holding labels fixed; the increment over trace-to-policy repair is 0.2 pp.

On the fixed unseen pool, verification rises by 5.1 pp as Brier loss changes from 0.194 to 0.164; the frozen policy changes only after its update, gaining 2.7 pp. Blocking the updated verifier's selection path reduces the next gain from 2.7 to 1.0 pp, while restoring that path from the same checkpoint yields 1.9 pp versus 0.5 pp when it remains blocked. Together, temporal separation and the block–restore contrast connect improved judgment to later policy learning. Appendix D.2 controls count, length, difficulty, and capacity; the feedback-withdrawal test in Appendix D extracts a further 1.8 pp from stored experience before the second plateau.

4.2.2 Does policy improvement supply useful new experience?

The reverse-path test asks whether policy improvement supplies more useful experience. With receiving verifiers, acquisition rules, tasks, and budgets fixed, the improved-policy source produces 5.3 rather than 2.6 pp on the Qwen tool task; matching source length, difficulty, and success rate retains 4.5 pp. The corresponding code advantage is only 0.2 pp, showing that the effect is not uniform across domains. Freezing only the experience generator leaves the final learner 3.7 pp behind the evolving-source branch; both learner policies continue updating, as detailed in Appendix D.3.

Holding confirmed successes fixed, new failures produce 3.2 pp of future policy gain versus 1.4 pp for matched old failures; when the common candidate pool contains no successful target, policy gain remains zero. Changed-condition prediction reaches 73.0% with local evidence, 62.0% without local execution, and 59.0% after shuffling action–outcome correspondences under the same token budget. The long-horizon difference is only 1.0 pp. These results connect better experience to improved judgment while delimiting the claim: the equal-information controls in Appendix D.3 test conditional transfer, not a complete causal world model.

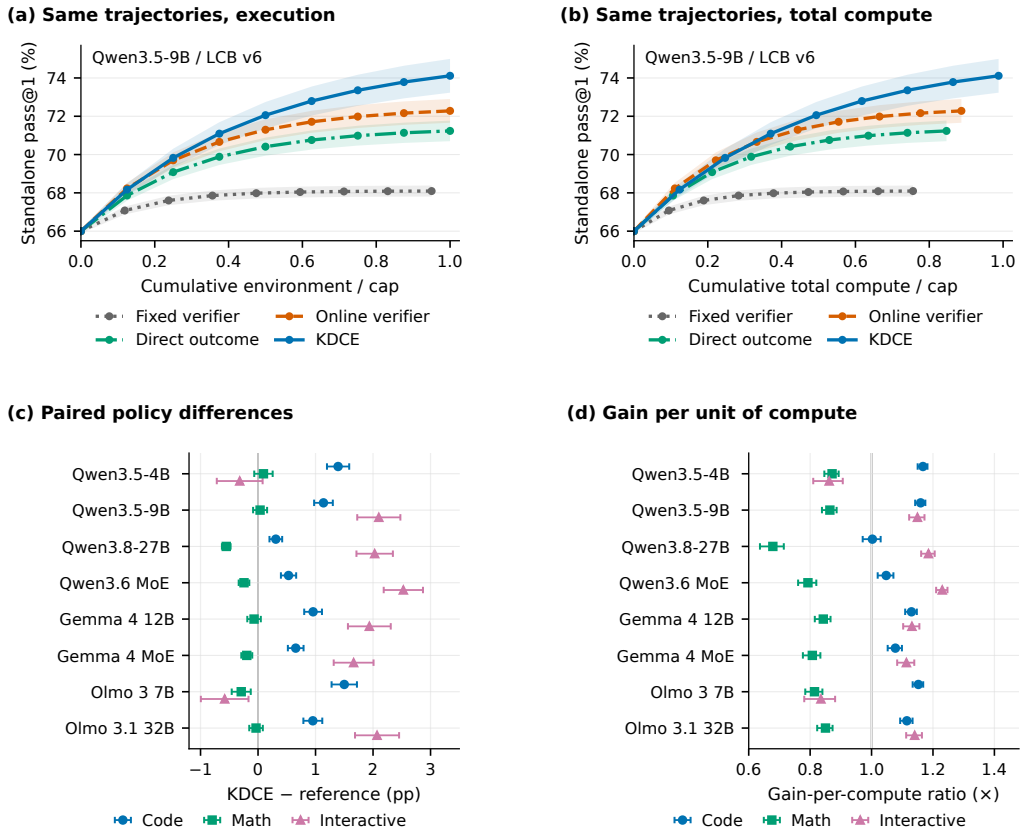

Figure 4: **Performance and efficiency.** Common-cap resource trajectories, paired policy differences, and gain per unit compute across model–domain settings.

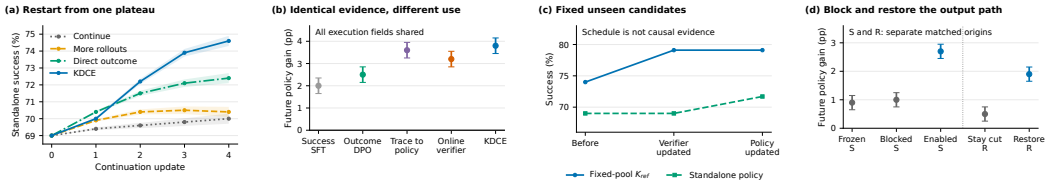

Figure 5: **Forward-path interventions.** Matched continuation, evidence use, fixed-pool verification, and block–restore tests; blue/green traces denote fixed-pool K_{ref} and standalone policy.

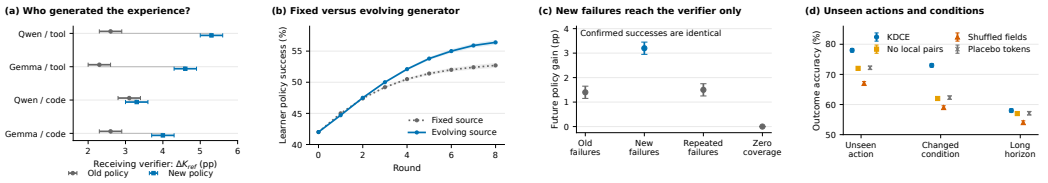

Figure 6: **Reverse-path interventions.** Experience-source, failure-routing, and unseen-action/condition probes.

4.2.3 Ablations and Limitations

Table 2 shows that the components contribute unevenly. Freezing verification and removing local executions reduce standalone performance by 5.2 and 1.8 pp; random queries cost only 0.3 pp,

Table 2: Qwen3.5-9B code ablations under common resource caps. D is the independent LCB endpoint; K_{ref} is forced selection on a separate fixed pool. Error, acceptance and correct yield are percentages from a one-candidate-per-original-task audit; yield equals acceptance times accepted correctness. Removed local-query budget is reallocated. The central sensitivity configuration is shown; risk and cost curves appear in Appendix E.

Variant	$D \uparrow$	$K_{\mathrm{ref}} \uparrow$	Error $\downarrow$	Accept	Correct yield	Compute
KDCE	74.1	82.7	3.0	72.0	69.8	0.988
Freeze verifier	68.9	73.8	8.0	61.0	56.1	0.760
No local executions	72.3	80.1	4.2	69.0	66.1	0.950
Random queries	73.8	82.3	3.3	71.0	68.7	0.960
Select with old verifier	71.9	82.7	6.8	68.0	63.4	0.980
No conservative selection	72.7	82.7	15.4	100.0	84.6	0.970
No ranking auxiliary	74.2	82.8	3.1	72.0	69.8	0.960

while removing the ranking auxiliary changes performance by $+0.1$ pp. Conservative selection lowers accepted error from 15.4% to 3.0% but also lowers correct yield from 84.6% to 69.8%. Equal-acceptance and nested-pool tests in Appendix E make this risk–coverage trade-off explicit.

Finally, the boundary analyses restrict the result to finite-budget settings with adequate coverage, useful feedback, and learnable targets. Old-rule performance drops 1.6 pp with replay and 8.0 pp without it; under distribution shift, accepted error reaches 14.0% at the original operating point. Appendix E.2 reports corrupted-feedback, low-absorption, forgetting, and cross-task robustness tests; complementary transfer, retrieval, update-schedule, and failure-routing controls appear in Appendices B.4, D.2, and D.3. Together, these failures bound the knowing–doing loop rather than supporting a universal improvement claim.

5 Related Work

Iterative self-improvement. STaR and ReST^{EM} bootstrap reasoning from self-generated solutions and correctness feedback (Zelikman et al., 2022; Singh et al., 2023). Generation–verification gap and sharpening analyses characterize how evaluative advantages support policy improvement (Song et al., 2025; Huang et al., 2025). KDCE extends this perspective by studying how action-derived knowledge renews these advantages across rounds.

Co-evolving evaluation and policies. Self-Rewarding and Meta-Rewarding improve generation and judgment together (Yuan et al., 2024; Wu et al., 2024). V-STaR learns verifiers from correct and incorrect solutions, while ReST-MCTS* and Sol-Ver couple solving with process rewards or generated tests (Hosseini et al., 2024; Zhang et al., 2024; Lin et al., 2025). STV uses reference-informed judgments to improve verification and guide policy learning (Wu & Raghunathan, 2026). SERPO co-evolves response evidence, rubrics, and policies, while RecurSE studies bounded judge–checker co-evolution (Wang et al., 2026; Liu et al., 2026). KDCE uses execution-supported behavioral knowledge and its applicability conditions to update evaluation under a bounded feedback budget.

Learning from grounded experience. Reflexion stores feedback-derived reflections, CodeRL learns an execution-supervised critic, and Absolute Zero uses executor feedback for task proposal and solving (Shinn et al., 2023; Le et al., 2022; Zhao et al., 2025). We build on grounded learning by using paired executions to form reusable, condition-dependent evaluation knowledge, and examine both directions of knowing–doing improvement through fixed-candidate, same-evidence, and directional interventions.

6 Conclusion

KDCE renews evaluative guidance by coupling budgeted execution, evidence-conditioned verification, uncertainty-aware target selection, and replayed policy updates. Matched directional controls test both feedback paths, while boundary analyses delimit the finite-budget regime in which the mechanism is expected to help.

Reproducibility Statement

Appendix C specifies model checkpoints, evaluation subsets, data boundaries, baselines, metrics, statistical units, and complete endpoint results. Appendix C.3 gives resource accounting and aggregation rules; Appendices D–E document directional controls, ablations, and operating boundaries. Appendix F provides round-wise pseudocode and numerical training, inference, calibration, and budget settings, while Appendix G states the assumptions and proofs for the conditional analysis. Core implementation code accompanies the submission.

AI Use Statement

Generative AI tools were used to assist with manuscript editing, coding, LaTeX formatting, source-comment translation, and visual layout. The authors take responsibility for the final content, claims, and artifacts.

References

Allen Institute for AI. Official Olmo 3 and Olmo 3.1 think model cards, 2026. URL https://huggingface.co/allenai/Olmo-3.1-32B-Think. Also Olmo-3-7B-Think. Accessed 2026-09-25.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d8e1344e27a5b08cdfd5d027d9b8d6de-Abstract.html.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. doi: 10.1093/biomet/26.4.404. URL https://academic.oup.com/biomet/article/26/4/404/291538.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs, 2026. URL https://arxiv.org/abs/2605.00674. Evaluation resources include AIME and February HMMT editions; accessed 2026-09-26.

EvalPlus Maintainers. EvalPlus: Official evaluation repository, 2026. URL https://github.com/evalplus/evalplus. Accessed 2026-09-25.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/4a8423d5e91fda00bb7e46540e2b0cf1-Paper.pdf.

Google. Official Gemma 4 model cards, 2026. URL https://huggingface.co/google/gemma-4-12B-it. Also gemma-4-26B-A4B-it. Accessed 2026-09-25.

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

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners, 2024. URL https://arxiv.org/abs/2402.06457.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745, 2011. URL https://arxiv.org/abs/1112.5745.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Audrey Huang, Adam Block, Dylan J. Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=WJaUkwci9o. Conference version; preprint posted in 2024.

Zixun Huang, Kishan Panaganti, Haitao Mi, and Leowei Liang. FlowBalance: Verifier-grounded self-improvement from on-policy reasoning experience, 2026. URL https://arxiv.org/abs/2609.03241. First posted September 3, 2026.

HuggingFaceH4. MATH-500 dataset card, 2026. URL https://huggingface.co/datasets/HuggingFaceH4/MATH-500. Accessed 2026-09-25.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

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. URL https://arxiv.org/abs/1612.01474.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. CodeRL: Mastering code generation through pretrained models and deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 21314–21328, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Zi Lin, Sheng Shen, Jingbo Shang, Jason Weston, and Yixin Nie. Learning to solve and verify: A self-play framework for code and test generation, 2025. URL https://arxiv.org/abs/2502.14948v2. Version 2.

Zi Lin, Sheng Shen, Ilia Kulikov, Jingbo Shang, Jason Weston, and Yixin Nie. Learning to solve and verify: A self-play framework for code and test generation. arXiv preprint arXiv:2502.14948v4, 2026. URL https://arxiv.org/abs/2502.14948v4. Version 4, March 2026; first posted February 2025.

Kaiyuan Liu, Ziyuan Zhuang, Rongxiang Weng, and Jieping Ye. RecurSE: Bounded recursive self-evaluation for LLM rubric judges, 2026. URL https://arxiv.org/abs/2608.24231.

LiveCodeBench Maintainers. LiveCodeBench: Official repository and versioned evaluation, 2026. URL https://github.com/LiveCodeBench/LiveCodeBench. release_v6; accessed 2026-09-25.

LiveCodeBench Pro Maintainers. LiveCodeBench Pro: Official evaluation tools, 2026. URL https://github.com/GavinZhengOI/LiveCodeBench-Pro. Accessed 2026-09-25.

Qwen Team. Official Qwen model cards, 2026. URL https://huggingface.co/Qwen/Qwen3.8-27B. Also Qwen3.5-4B, Qwen3.5-9B and Qwen3.6-35B-A3B. Accessed 2026-09-25.

Scale AI. SWE-bench Pro: Public dataset and evaluation resources, 2026. URL https://huggingface.co/datasets/ScaleAI/SWE-bench_Pro. Reported runs use the preserved v1 public configuration and a frozen pre-V2 study subset; accessed 2026-09-26.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, et al. Beyond human data: Scaling self-training for problem-solving with language models, 2023. URL https://arxiv.org/abs/2312.06585.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n.

Yuda Song, Hanlin Zhang, Carson Eisenach, Sham M. Kakade, Dean Foster, and Udaya Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mtJSMcF3ek. arXiv version 2.

Yifan Sun, Yushan Liang, Zhen Zhang, Xin Liu, and Jiaye Teng. Theoretical modeling of large language model self-improvement training dynamics through solver-verifier gap. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1f6100363156cced8633f4e89dd8ceb1-Paper-Conference.pdf.

Terminal-Bench Team. Terminal-bench 4.0, 2026. URL https://www.tbench.ai/news/terminal-bench-4-0. Official release announcement; accessed 2026-09-25; version 4.0.0.

Jianze Wang, Kunwang Zheng, Ying Liu, Yu Cao, Qilong Zhang, Jinlong Chen, Hua Yang, and Qianglong Chen. SERPO: Self-evolving rubric policy optimization for open-ended test-time reinforcement learning, 2026. URL https://arxiv.org/abs/2607.26873.

Chen Henry Wu and Aditi Raghunathan. Self-trained verification for training- and test-time self-improvement, 2026. URL https://arxiv.org/abs/2605.30290. arXiv preprint.

Tianhao Wu et al. Meta-rewarding language models: Self-improving alignment with LLM-as-a-meta-judge, 2024. URL https://arxiv.org/abs/2407.19594.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2024. URL https://arxiv.org/abs/2401.10020.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search, 2024. URL https://arxiv.org/abs/2406.03816.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.15240.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, et al. Absolute zero: Reinforced self-play reasoning with zero data, 2025. URL https://arxiv.org/abs/2505.03335.

A EXPERIMENTAL SETUP AND MEASUREMENT

The appendix proceeds from shared setup and measurement to observation diagnostics (Appendix B), complete performance and resources (Appendix C), directional controls (Appendix D), reliability and boundaries (Appendix E), implementation (Appendix F), and conditional analysis (Appendix G). All accuracy differences are in percentage points (pp); configuration sensitivity is distinct from task-sampling uncertainty.

A.1 MODELS, TASKS, AND ADMISSIBLE INFORMATION

The text-only matrix covers eight checkpoints from Qwen, Gemma, and Olmo and eleven evaluations (Table 3) (Qwen Team, 2026; Google, 2026; Allen Institute for AI, 2026). Architecture, size, pretraining, post-training, and reasoning format vary jointly; cross-model differences are not causal scaling estimates. Native chat templates and reasoning settings are fixed within each checkpoint. Standalone code/math evaluation uses one greedy generation; interaction uses one tool-enabled episode under fixed token, action, and wall-time limits, without learned-verifier selection or repair. Truncations and valid task failures remain in the denominator.

Training uses separate executable programs and tests, mathematical problem families, repositories, and resettable tool tasks. Original tasks and all paraphrases, edits, traces, and condition variants stay in the same partition. Deduplication covers normalized text, available solution/test signatures, repository ancestry, and cross-benchmark overlap. Development, temperature fitting, threshold calibration, clean-update diagnostics, and final audits are separate from gradient/replay data; final outcomes never select hyperparameters, sources, or stopping points. Mathematical step labels require an actual symbolic/formal check; code feedback tests are distinct from held-out tests. Tools use versioned interfaces, reset states, success predicates, and fixed timeout/retry rules. For SWE-bench Pro, all reported runs use the preserved v1 public configuration and a frozen study subset fixed before the V2 benchmark update.

Learners receive only queried outcomes and context available before the predicted action. Hidden tests, reference answers, unqueried outcomes, and oracle mechanism labels are excluded from routine training. Generated claims of success are not execution evidence. Records preserve task/parent/query IDs, split, checkpoint, candidate hash, decoding/seed, environment version, observed-field masks, costs, and downstream use. Infrastructure failures are logged separately and charged; oracle insertion and sealed audit labels remain diagnostic-only.

A.2 BASELINE IMPLEMENTATIONS AND SUPERVISION TIERS

All matched-feedback methods start from the same solver weights and share the task-normalized policy update and resource caps. **Fixed verifier** freezes the initialized scorer; **success filtering** retains execution-confirmed positives (ReST-EM-style) (Singh et al., 2023); **direct outcome** uses queried binary rewards, within-task centered advantages, and policy KL without a learned verifier. The direct learner may reallocate verification costs to valid rollouts or updates. Initialization, reference-policy computation, and development are charged; scheduled prefixes are not independently tuned budget optima.

The **online verifier** matches KDCE's backbone, ensemble, observed terminal/local-field targets, calibration, conservative selector, and policy updater. It queries randomly and learns records independently. KDCE additionally uses active acquisition, executed local comparisons, and within-task ranking. Both receive all recorded local-edit fields in fixed-log comparisons. **V-STaR-style (adapt.)** trains preferences from observed positive/negative pairs with the same policy updater, but no reference-solution seed or deployment Best-of-N; it is an objective-level adaptation, not a reproduction of the original deployment protocol (Hosseini et al., 2024).

SOL-VER generates tests and code (Lin et al., 2026, v4); ReST-MCTS* includes process-reward search and target construction (Zhang et al., 2024). These remain separate end-to-end comparisons. ViL + STV uses reference-solution-assisted verifier supervision, whereas ViL + self-verify does not (Wu & Raghunathan, 2026). Self-Rewarding and Meta-Rewarding already study joint generation/evaluation learning (Yuan et al., 2024; Wu et al., 2024). Our controls isolate evidence use and resource allocation rather than superiority over all original systems.

Table 3: Model identities and evaluation subsets. Checkpoint-card scores are not controlled results. Benchmark sources: (EvalPlus Maintainers, 2026; LiveCodeBench Maintainers, 2026; LiveCodeBench Pro Maintainers, 2026; HuggingFaceH4, 2026; Dekoninck et al., 2026; Scale AI, 2026; Terminal-Bench Team, 2026).

(a) Model checkpoints

Display name	Checkpoint identifier	Architecture
Qwen3.5-4B	`Qwen/Qwen3.5-4B`	dense
Qwen3.5-9B	`Qwen/Qwen3.5-9B`	dense
Qwen3.8-27B	`Qwen/Qwen3.8-27B`	dense, hybrid attention
Qwen3.6 MoE	`Qwen/Qwen3.6-35B-A3B`	MoE
Gemma 4 12B	`google/gemma-4-12B-it`	dense
Gemma 4 MoE	`google/gemma-4-26B-A4B-it`	MoE
Olmo 3 7B	`allenai/Olmo-3-7B-Think`	dense
Olmo 3.1 32B	`allenai/Olmo-3.1-32B-Think`	dense

(b) Evaluation matrix

Evaluation	Domain	Version / subset
HumanEval+	Code	HumanEval+ frozen subset
MBPP+	Code	Sanitized MBPP+ frozen subset
LiveCodeBench v6	Code	`release_v6`, cumulative subset
LiveCodeBench Pro	Code	Public C++ judging subset
MATH-500	Math	Fixed test split
AIME 2025	Math	2025 I and II
AIME 2026	Math	2026 I and II
HMMT Feb 2025	Math	February 2025
HMMT Feb 2026	Math	February 2026
SWE-bench Pro Public v1	Interactive	Frozen pre-V2 public subset
Terminal-Bench 4.0	Interactive	Version 4.0.0 subset

A.3 Metrics, evaluation distributions, and statistical units

For task $x \sim \mathcal{T}$, fixed-protocol candidates $\tau_{1:N} \sim \pi_t$, and binary audit outcome y, let $A = \{\max_i y(x, \tau_i) = 1\}$. The verifier selects exactly one candidate $\hat{\tau}$ with a fixed tie rule:

$$D = \mathbb{E}[y(x, \tau_1)], \quad C = \Pr(A), \quad R = \Pr(y(x, \hat{\tau}) = 1 \mid A), \quad K = \Pr(y(x, \hat{\tau}) = 1). \tag{8}$$

Then $K = CR$ and $G = K - D$. When $C = 0$, $K = 0$ and R is undefined; at $N = 1$, $K = D$ and $R = 1$ when $C > 0$, without implying zero learning gain. Estimate C over all pools, R over success-containing pools, K over all selections, and D from a predetermined pool index or an independent draw with the same marginal protocol. The empirical identity is exact before rounding only with common denominators and weights. Compute $K_d = C_d R_d$ before domain aggregation: $K_{\text{macro}} = \sum_d w_d C_d R_d$, not the product of separately averaged columns. Continuous utility requires direct expectation differences, not the binary factorization.

Three distributions remain distinct. Fixed reference pools freeze tasks, candidate text, and labels across verifier updates and support K_{ref}, NLL, Brier score, and available localization/counterfactual metrics. Same-current-policy pools compare old/new scorers at fixed coverage:

$$K(\pi_t, v_{t+1}; N) - K(\pi_t, v_t; N) = C_t(N)\big[R(\pi_t, v_{t+1}; N) - R(\pi_t, v_t; N)\big]. \tag{9}$$

Fresh post-update pools measure online joint K_t and standalone D_t. A frozen verifier can therefore have changing online K. Future outcomes are targets, never predictor inputs. For abstaining selection, report acceptance a, conditional error e, and per-task correct yield $a(1 - e)$ separately from forced-choice K; multiple confirmed positives and replay are not counted by this one-candidate yield.

Independent units are parent tasks, repositories, or problem families. Paired branches share checkpoints and candidate banks; repeated completions do not create independent competition problems.

Endpoint tables report central values and half-ranges across five paired sensitivity configurations, not sampling confidence intervals. Task-sampling uncertainty is clustered at the relevant parent unit. Small differences do not establish equivalence. Family-balanced aggregation and efficiency are specified in Appendix C.3.

A.4 Independent uptake and resource-matched comparisons

Estimate update effectiveness U with policy clones trained on an identical verified calibration batch, then evaluated on disjoint eligible variants using the fixed recipe:

$$\widehat{U} = \frac{1}{m} \sum_{j=1}^{m} y(x_j^{\text{variant}}, \tau_j^{\text{updated}}), \qquad \widehat{S} = \widehat{U}\,\widehat{G}. \tag{10}$$

Here m counts eligible held-out variants, not training successes. Compare a no-update clone and report retention; U is undefined for an empty eligible set. It is neither $\Delta D/G$ nor the method's total-variation error. Fit $\mathbb{E}[\Delta D_t] \approx \beta U_t G_t + b$ only on calibration trajectories, freeze coefficients, and evaluate held-out rounds, families, models, and domains against G, C, N, D and training loss. Cross-fit baseline estimates to avoid shared measurement error in G and ΔD.

Compute-matched and environment-call-matched comparisons are separate, with unmatched resources disclosed (Snell et al., 2025). Charge generation, all verifier scoring, both learners' updates, edits, calibration, retrieval/source trials, retries, and initialization. Multi-field returns cost one call; reruns cost additional calls. Sealed audit cost is separate. Same-log branches use identical immutable records and authorized fields; online policies may collect different trajectories within matched interfaces and caps. Unused budget is reported, not consumed by meaningless calls.

B Observation diagnostics: stagnation, feedback, and transfer

B.1 Training dynamics, residual tasks, and sampling

The main diagnostic uses $N = 64$ and a fixed held-out task panel. Table 4(a) tracks $D, C, R, K, G, U, S = UG$ and subsequent gain; it is distinct from the isolated update branches in panel (b). A separate eligibility batch defines each checkpoint's remaining unsolved tasks; fresh pools estimate their metrics without feeding the audit-defined residual set into the training curriculum. Residual coverage falls from 46.2% to 24.7% and conditional recognition from 63.5% to 41.8%, on a changing task distribution.

Among prespecified diagnostic families failing after an update, sequential categories are no correct candidate (48%), a missed available correct candidate (34%), and selected-correct but failed held-out transfer (18%). Infrastructure-invalid cases are outside this denominator. These are operational audit categories, not identified optimization causes.

At fixed checkpoints, prompts, and decoding, sample the largest pool once, randomize order, and use nested prefixes at $N = 1, 4, 16, 64, 256$. Downstream clones share update tokens and steps; generation/scoring costs are recorded separately. Coverage improves while selection can deteriorate. Audits of selected-label error, confidence, rejection yield, and retention test whether ranking errors propagate through training. The zero update gain at $N = 1$ is observed, not implied by $K = D$.

A paired insertion audit replaces, rather than adds, one candidate with an independently checked success; controls match length/style and pool size. Top-1 recognition is 81.6% on ordinary tasks and 37.9% on remaining unsolved tasks, with 61.4% top-5 recognition in the latter. Inserted trajectories remain diagnostic-only. Thus success availability and usable guidance are distinct.

B.2 Response prediction and matched interventions

Using Appendix A.4's frozen calibration-to-test pipeline, UG predicts next-round gain better than G alone in Table 5(a). Its $R^2 = 0.83$ is a held-out association, not a universal law or a consequence of solver–verifier theory (Sun et al., 2026). Calibration coefficients, residual checks, and alternative predictors are not fit on the outcomes being predicted.

Table 4: Stagnation and sampling diagnostics. Accuracy columns are percentages; gaps and policy gains are pp, and U is dimensionless. Panel (a) is the longitudinal run; panel (b) uses independent matched-update branches. Rounded macro columns must not be multiplied.

(a) Training dynamics

t	D	C	R	K	G	U	S	ΔD
0	31.8	68.1	77.0	52.4	20.6	.72	14.8	8.8
1	40.6	70.2	75.6	53.1	12.5	.70	8.8	6.4
2	47.0	71.0	74.2	52.7	5.7	.68	3.9	3.1
3	50.1	70.7	73.6	52.0	1.9	.66	1.3	0.3
4	50.4	70.4	73.4	51.7	1.3	.65	0.8	0.1

(b) Nested rollout-count sweep

N	C	R	K	D	G	ΔD
1	50.4	100.0	50.4	50.4	0.0	0.0
4	57.8	88.1	50.9	50.4	0.5	0.2
16	66.5	78.2	52.0	50.4	1.6	0.8
64	70.4	73.4	51.7	50.4	1.3	0.5
256	72.0	68.5	49.3	50.4	-1.1	-0.7

On identical $N = 64$ pools, replace only the selector with oracle, external, self-verifier, or random selection, then train matched diagnostic clones. High-confidence error is the incidence $\Pr(y = 0, q \geq h)$ over all selected candidates, distinct from ordinary error $1 - K$ and conditional tail risk. Oracle/random selectors lack native learned confidence; random ordinary error is 49.6%.

The factorial intervention separately changes candidate composition, ranking, and the update protocol; U is independently recalibrated, and C, R are remeasured because support changes can affect ranking difficulty. The shared baseline is $D = 0.504$. Large positive G can coexist with weak policy uptake, consistent with the need to test transfer beyond fitted responses (Kumar et al., 2025). Oracle replacement and clean-target updates are upper-bound diagnostics, not outcome-only training. Algebraic identities and selection-regret bounds do not independently validate the learning mechanism (Appendix G).

B.3 Feedback content and action-sensitive generalization

Six feedback representations are derived from the same trajectory log and verifier initialization (Table 6). Fixed unseen reference pools measure verifier change; separate policy clones learn the resulting selections and are evaluated on unseen variants. Common-log, token-matched, and call-matched comparisons remain distinct. Outcome-trained generative verifiers and unit-test critics motivate the outcome-only baseline (Zhang et al., 2025; Le et al., 2022). Checked explanations are audited against actual outputs/transitions; uncheckable claims remain unverified, and every checking call is charged.

Counterfactual probes reset held-out states, change one valid action, and obtain labels by execution or a validated local checker; stochastic transitions use charged replications. Checked explanations improve counterfactual accuracy by 10.3 pp versus 2.1 pp for outcome-only feedback. With final labels fixed, replacing supported explanations by incorrect causes reduces fixed-reference gain from 8.2 to 1.0 pp and yields a related-task change of -0.8 pp. Length-/format-matched irrelevant explanations and neutral field masks separate information content from fluency and text length. Training-time masking tests learning value; evaluation-time masking tests dependence on available inputs.

Independent calibration loss reduction is compared with length, fluency, difficulty, and source identity on held-out families. Its rank correlation with verification improvement is 0.74; it is not measured mutual information or a causal effect. The ideal Bayes log-loss identity and its approximation-error qualification appear in Appendix G.3. Exact/isomorphic deduplication and task-family splits prevent trace leakage. Qualitative records retain state, action, consequence, proposed cause/alternative, supporting checks, and applicability conditions; an unexecuted alternative is not established knowledge.

Table 5: Complementary diagnostics. (a) Held-out response prediction. (b) Same-pool selector replacement; oracle and random confidence entries are inapplicable. (c) Matched composition, recognition, and uptake interventions, with $G = CR - D$.

(a) Predicting subsequent policy gain

Predictor	Held-out R^2	Spearman ρ
Rollout count N	.07	.18
Current accuracy D_t	.16	-.31
Training loss	.22	.39
Coverage C_t	.35	.55
Selection advantage G_t	.71	.84
Uptake-adjusted advantage $U_t G_t$	.83	.91

(b) Selector replacement at fixed coverage

Selector	Selected@64 (%)	Update gain (pp)	High-confidence error (%)
Oracle	70.4	4.1	—
External verifier	64.8	3.0	4.7
Self-verifier	51.7	0.5	15.9
Uniform random	50.4	0.0	—

(c) Candidate and update interventions

Condition	C	R	U	G (pp)	ΔD (pp)
Lower coverage	.57	.90	.75	0.9	0.4
Lower recognition	.70	.70	.75	-1.4	-0.9
Lower update effectiveness	.70	.90	.15	12.6	0.8
Higher settings	.70	.90	.75	12.6	5.7

Table 6: Feedback representation on common trajectories. The first four numeric columns are gains in pp; confident error is the fraction of elicited explanations above the preset confidence threshold and contradicted by evidence.

Feedback	ΔK_{ref}	Error location	Counter-factual	Variant ΔD	Confident error (%)
Trajectory only	0.3	0.2	0.1	0.1	18.7
Unchecked reflection	0.8	1.0	0.6	0.3	22.4
Final outcome	2.6	1.8	2.1	1.2	12.1
Localized feedback	5.5	7.3	5.0	2.9	7.4
State–action–outcome	6.9	6.8	9.1	3.6	5.8
Checked explanation	8.2	9.4	10.3	4.5	3.1

B.4 Cross-task reuse and finite-budget generalization

Strict leave-one-task-out transfer excludes target outcomes from training, calibration, source selection, stopping, and development. Frozen source-trained verifiers select target-policy training data without target execution feedback; final target checks are sealed. The observed differences are approximately 4–5 pp in verification and 1–2 pp in behavior. Separate source clones are evaluated on all eligible target reference pools; diagonal same-task results are not transfer. Across matched source–target pairs, correlations with transfer are 0.68 for primitive overlap, 0.76 for the information proxy, and 0.29 for textual similarity; they are associations, not information laws.

Table 7 instead permits limited target-development feedback. Selective reuse filters incompatible versions/conditions, ranks eligible records with a frozen representation, and compares source-augmented versus target-only clones by permitted development loss; trial updates and source collection are charged. Oracle source selection has extra-information access and remains separate. Negative transfer is the fraction of target tasks whose K_{ref} falls below the pre-update value by the preset tolerance. Global pooling's 14.8% falls to 4.1% with selective reuse, but harm remains. Target-call ratios are relative to target-only at unequal achieved performance; they are neither total-cost nor equal-performance savings.

Held-out combinations, states, and preconditions exclude all related variants before learning. Predictors see the proposed action and preceding context, never its consequence; invalid counterfactuals are recorded separately. To connect judgment with behavior, compare old/new verifiers on identical reference and current-policy pools, then train matched policy clones without changing the originating feedback permissions.

A separate blinded semantic audit by doctoral-student researchers compares successful updated trajectories with a fixed finite base pool, using algorithmic structure, mathematical routes, or meaningful state transitions rather than string novelty. Strategies absent from that pool account for 4.2% of audited self-labeling successes and 21.6% of selective-update successes; ambiguous matches remain separate. This motivates trajectory-level analysis alongside related outcome-grounded work (Huang et al., 2026). Finite non-observation does not establish zero base-policy probability or support expansion.

Table 7: Limited-target-feedback reuse comparison. Target-call ratios are distinct from total system cost and do not establish equal-performance savings.

Experience source	ΔK_{ref} (pp)	ΔD (pp)	Negative transfer (%)	Target-call ratio
No additional experience	0.0	0.0	0.0	—
Target only	5.8	3.0	3.2	1.00
Global pool	2.4	1.0	14.8	0.70
Text retrieval	4.1	1.9	8.7	0.62
Selective reuse	6.7	3.4	4.1	0.48
Oracle selection	7.5	3.9	1.9	0.43

B.5 Eight-round learning and feedback dose

The six systems share initial checkpoints, splits, and an eight-round schedule: unchecked self-labeling, more sampling with a frozen verifier, same-task feedback, global pooling, selective updates, and a privileged oracle branch. Table 8(a) displays alternate rounds; labels from oracle selection never enter ordinary systems. Self-labeling plateaus around round four, whereas selective updates continue improving over this finite window. These controlled systems are not reproductions of all neighboring methods (Appendix A.2).

Panel (b) varies the fraction of proposed rollouts receiving grounded feedback while holding the exposure rule and proposal count fixed. The remaining rollouts obtain no new execution label; call-matched online performance is a separate comparison. Online D, C, R, K, G, accepted-label error, yield, and retention are distinct from fixed-pool verifier audits. The dose points establish no critical feedback threshold. Same-pool replacement and later policy effects are tested in Appendix D; corruption, delay, forgetting, and source interactions are consolidated in Appendix E.2.

Table 8: Multi-round outcomes and feedback dose. (a) Accuracies are percentages, G_8 is pp, and label error is audited over accepted training labels. (b) Round-eight standalone success at each grounded-rollout fraction. Figure 2 summarizes these results.

(a) Six systems over eight rounds

System	D_0	D_2	D_4 (%)	D_6	D_8	G_8 (pp)	Label error (%)
Self-labeling	31.8	47.0	50.4	50.6	50.5	0.7	16.8
More sampling	31.8	47.5	51.0	50.6	49.9	-0.6	19.4
Same-task feedback	31.8	48.4	53.7	56.2	57.8	3.2	6.3
Global pooling	31.8	49.0	55.1	57.2	58.4	2.7	9.8
Selective updates	31.8	49.6	56.8	60.1	62.7	5.4	4.5
Oracle updates	31.8	51.2	59.5	64.0	66.3	7.1	0.0

(b) Grounded feedback proportion

Grounded rollout fraction (%)	0	1	2.5	5	10	20
Round-eight D (%)	50.5	51.3	53.9	58.8	62.7	63.4

C Complete performance and resource results

C.1 Complete endpoints and published context

Tables 9 and 10 retain every model/method/task endpoint and its configuration half-range. Code and interaction share one presentation table, not a training run or evaluation protocol. Parent-level uncertainty and benchmark-family weighting follow Appendix A.3. The controlled values are never replaced by model-card or source-paper scores.

Table 9: Complete code and interactive endpoints (%). Entries are central value ± half-range across five paired configurations. SWE Pro v1 and Terminal use standalone tool-enabled episodes, not verifier-assisted inference.

Method	HE+	MBPP+	LCB v6	LCB Pro	SWE Pro v1	Terminal
Qwen3.5-4B						
Base	88.0 ± 0.0	75.0 ± 0.1	52.0 ± 0.2	5.0 ± 0.3	13.0 ± 0.3	7.0 ± 0.3
Fixed verifier	88.7 ± 0.1	76.1 ± 0.2	54.6 ± 0.4	6.2 ± 0.4	14.7 ± 0.4	8.5 ± 0.4
Success filtering	89.0 ± 0.1	76.7 ± 0.2	55.9 ± 0.5	6.7 ± 0.5	16.3 ± 0.6	10.1 ± 0.6
Direct outcome	89.7 ± 0.2	77.8 ± 0.3	58.4 ± 0.7	7.9 ± 0.6	19.6 ± 0.8	13.1 ± 0.8
Online verifier	90.0 ± 0.2	78.4 ± 0.4	59.7 ± 0.8	8.5 ± 0.6	21.3 ± 1.0	14.7 ± 0.9
V-STaR-style (adapt.)	89.9 ± 0.2	78.3 ± 0.3	59.4 ± 0.8	8.3 ± 0.6	20.5 ± 0.9	13.9 ± 0.9
KDCE	90.6 ± 0.3	79.4 ± 0.5	62.0 ± 1.1	9.5 ± 0.8	20.9 ± 1.4	14.4 ± 1.3
Qwen3.5-9B						
Base	92.0 ± 0.0	84.0 ± 0.1	66.0 ± 0.1	12.0 ± 0.3	23.0 ± 0.3	14.0 ± 0.3
Fixed verifier	92.5 ± 0.1	84.9 ± 0.1	68.1 ± 0.3	13.1 ± 0.4	24.5 ± 0.4	15.5 ± 0.4
Success filtering	92.8 ± 0.1	85.3 ± 0.2	69.1 ± 0.4	13.7 ± 0.4	26.1 ± 0.5	16.9 ± 0.5
Direct outcome	93.3 ± 0.1	86.2 ± 0.2	71.2 ± 0.5	14.8 ± 0.5	29.2 ± 0.8	19.8 ± 0.8
Online verifier	93.6 ± 0.2	86.6 ± 0.3	72.3 ± 0.6	15.3 ± 0.6	30.7 ± 0.9	21.3 ± 0.9
V-STaR-style (adapt.)	93.5 ± 0.1	86.5 ± 0.3	72.0 ± 0.6	15.2 ± 0.6	29.9 ± 0.8	20.6 ± 0.8
KDCE	94.0 ± 0.2	87.4 ± 0.4	74.1 ± 0.9	16.3 ± 0.7	32.8 ± 1.3	23.4 ± 1.3
Qwen3.8-27B						
Base	97.0 ± 0.0	91.0 ± 0.0	86.0 ± 0.0	30.0 ± 0.2	46.0 ± 0.2	30.0 ± 0.2
Fixed verifier	97.3 ± 0.0	91.6 ± 0.1	87.2 ± 0.1	31.0 ± 0.3	47.2 ± 0.3	31.3 ± 0.3
Success filtering	97.4 ± 0.0	91.9 ± 0.1	87.8 ± 0.2	31.5 ± 0.4	48.5 ± 0.4	32.6 ± 0.5
Direct outcome	97.7 ± 0.1	92.5 ± 0.2	89.1 ± 0.3	32.4 ± 0.4	51.0 ± 0.6	35.2 ± 0.7
Online verifier	97.9 ± 0.1	92.8 ± 0.2	89.7 ± 0.3	32.9 ± 0.5	52.2 ± 0.7	36.5 ± 0.8
V-STaR-style (adapt.)	97.8 ± 0.1	92.8 ± 0.2	89.5 ± 0.3	32.8 ± 0.5	51.6 ± 0.6	35.8 ± 0.7
KDCE	98.0 ± 0.1	93.1 ± 0.3	90.1 ± 0.5	33.2 ± 0.6	54.2 ± 1.0	38.5 ± 1.1
Qwen3.6 MoE						
Base	95.0 ± 0.0	89.0 ± 0.0	78.0 ± 0.1	23.0 ± 0.3	37.0 ± 0.2	23.0 ± 0.3
Fixed verifier	95.4 ± 0.0	89.7 ± 0.1	79.6 ± 0.2	24.0 ± 0.4	38.4 ± 0.3	24.4 ± 0.4
Success filtering	95.6 ± 0.1	90.0 ± 0.1	80.4 ± 0.3	24.5 ± 0.4	39.7 ± 0.4	25.7 ± 0.5
Direct outcome	96.0 ± 0.1	90.7 ± 0.2	82.0 ± 0.4	25.6 ± 0.5	42.5 ± 0.7	28.5 ± 0.7
Online verifier	96.2 ± 0.1	91.1 ± 0.2	82.8 ± 0.5	26.1 ± 0.5	43.8 ± 0.8	29.8 ± 0.8
V-STaR-style (adapt.)	96.1 ± 0.1	91.0 ± 0.2	82.6 ± 0.4	25.9 ± 0.5	43.1 ± 0.7	29.2 ± 0.8
KDCE	96.4 ± 0.2	91.4 ± 0.3	83.6 ± 0.7	26.6 ± 0.6	46.3 ± 1.1	32.4 ± 1.2
Gemma 4 12B						
Base	93.0 ± 0.0	85.0 ± 0.1	70.0 ± 0.1	13.0 ± 0.3	24.0 ± 0.3	15.0 ± 0.3
Fixed verifier	93.5 ± 0.1	85.8 ± 0.1	71.9 ± 0.3	14.1 ± 0.4	25.5 ± 0.4	16.5 ± 0.4
Success filtering	93.7 ± 0.1	86.2 ± 0.2	72.9 ± 0.3	14.7 ± 0.4	27.1 ± 0.5	17.9 ± 0.5
Direct outcome	94.2 ± 0.1	87.1 ± 0.2	74.9 ± 0.5	15.8 ± 0.5	30.1 ± 0.8	20.8 ± 0.8
Online verifier	94.5 ± 0.1	87.5 ± 0.3	75.8 ± 0.6	16.3 ± 0.6	31.6 ± 0.9	22.3 ± 0.9
V-STaR-style (adapt.)	94.4 ± 0.1	87.4 ± 0.2	75.6 ± 0.6	16.2 ± 0.6	30.9 ± 0.8	21.5 ± 0.8
KDCE	94.8 ± 0.2	88.1 ± 0.4	77.3 ± 0.8	17.2 ± 0.7	33.6 ± 1.3	24.1 ± 1.2
Gemma 4 MoE						
Base	96.0 ± 0.0	89.0 ± 0.0	76.0 ± 0.1	19.0 ± 0.3	34.0 ± 0.2	23.0 ± 0.3
Fixed verifier	96.3 ± 0.0	89.7 ± 0.1	77.7 ± 0.2	20.1 ± 0.4	35.4 ± 0.3	24.4 ± 0.4
Success filtering	96.5 ± 0.1	90.0 ± 0.1	78.5 ± 0.3	20.6 ± 0.4	36.8 ± 0.5	25.7 ± 0.5
Direct outcome	96.9 ± 0.1	90.7 ± 0.2	80.2 ± 0.4	21.6 ± 0.5	39.6 ± 0.7	28.5 ± 0.7
Online verifier	97.0 ± 0.1	91.1 ± 0.2	81.1 ± 0.5	22.2 ± 0.5	41.0 ± 0.8	29.8 ± 0.8
V-STaR-style (adapt.)	97.0 ± 0.1	91.0 ± 0.2	80.9 ± 0.5	22.0 ± 0.5	40.3 ± 0.7	29.2 ± 0.8
KDCE	97.3 ± 0.1	91.5 ± 0.3	82.1 ± 0.7	22.8 ± 0.7	42.7 ± 1.1	31.5 ± 1.2

Method	HE+	MBPP+	LCB v6	LCB Pro	SWE Pro v1	Terminal
Olmo 3 7B						
Base	84.0 ± 0.1	69.0 ± 0.1	36.0 ± 0.2	1.5 ± 0.3	6.0 ± 0.3	2.0 ± 0.3
Fixed verifier	84.8 ± 0.1	70.3 ± 0.2	39.1 ± 0.5	2.7 ± 0.4	7.7 ± 0.5	3.6 ± 0.5
Success filtering	85.2 ± 0.2	70.9 ± 0.3	40.6 ± 0.6	3.3 ± 0.5	9.5 ± 0.6	5.2 ± 0.6
Direct outcome	86.0 ± 0.2	72.2 ± 0.4	43.7 ± 0.8	4.5 ± 0.6	12.9 ± 0.9	8.3 ± 0.8
Online verifier	86.4 ± 0.2	72.9 ± 0.4	45.2 ± 1.0	5.1 ± 0.6	14.7 ± 1.0	9.9 ± 1.0
V-STaR-style (adapt.)	86.3 ± 0.2	72.7 ± 0.4	44.8 ± 0.9	4.9 ± 0.6	13.8 ± 1.0	9.1 ± 0.9
KDCE	87.1 ± 0.3	74.0 ± 0.6	47.8 ± 1.3	6.1 ± 0.8	14.1 ± 1.5	9.4 ± 1.4
Olmo 3.1 32B						
Base	94.0 ± 0.0	86.0 ± 0.0	63.0 ± 0.1	11.0 ± 0.3	20.0 ± 0.3	11.0 ± 0.3
Fixed verifier	94.4 ± 0.1	86.8 ± 0.1	65.2 ± 0.3	12.1 ± 0.4	21.6 ± 0.4	12.5 ± 0.4
Success filtering	94.7 ± 0.1	87.2 ± 0.1	66.3 ± 0.4	12.7 ± 0.4	23.1 ± 0.5	14.0 ± 0.6
Direct outcome	95.1 ± 0.1	88.0 ± 0.2	68.5 ± 0.6	13.8 ± 0.5	26.3 ± 0.8	17.0 ± 0.8
Online verifier	95.3 ± 0.1	88.4 ± 0.2	69.6 ± 0.7	14.4 ± 0.6	27.9 ± 0.9	18.5 ± 0.9
V-STaR-style (adapt.)	95.3 ± 0.1	88.3 ± 0.2	69.3 ± 0.6	14.2 ± 0.6	27.1 ± 0.8	17.7 ± 0.8
KDCE	95.7 ± 0.2	89.0 ± 0.3	71.2 ± 0.9	15.2 ± 0.7	30.0 ± 1.3	20.5 ± 1.3

Table 10: Complete mathematics endpoints (%), with the same five-configuration half-range convention. Competition scores are aggregate rates, not counts of independent solved problems.

Method	MATH-500	AIME 25	AIME 26	HMMT 25	HMMT 26
Qwen3.5-4B					
Base	82.0 ± 0.1	52.0 ± 0.2	43.0 ± 0.2	35.0 ± 0.2	28.0 ± 0.3
Fixed verifier	82.5 ± 0.1	53.6 ± 0.3	44.9 ± 0.4	36.9 ± 0.4	30.1 ± 0.4
Success filtering	82.8 ± 0.1	54.7 ± 0.4	46.2 ± 0.5	38.1 ± 0.5	31.5 ± 0.5
Direct outcome	83.9 ± 0.2	58.6 ± 0.7	50.7 ± 0.8	42.4 ± 0.8	36.4 ± 0.9
Online verifier	83.6 ± 0.2	57.5 ± 0.6	49.4 ± 0.7	41.2 ± 0.7	35.0 ± 0.8
V-STaR-style (adapt.)	83.8 ± 0.2	58.0 ± 0.6	50.1 ± 0.8	41.8 ± 0.8	35.7 ± 0.9
KDCE	84.0 ± 0.3	58.7 ± 0.9	50.8 ± 1.0	42.5 ± 1.0	36.5 ± 1.2
Qwen3.5-9B					
Base	91.0 ± 0.0	72.0 ± 0.1	66.0 ± 0.1	56.0 ± 0.2	47.0 ± 0.2
Fixed verifier	91.3 ± 0.1	73.2 ± 0.2	67.4 ± 0.2	57.5 ± 0.3	48.7 ± 0.3
Success filtering	91.5 ± 0.1	74.0 ± 0.3	68.4 ± 0.3	58.4 ± 0.3	49.9 ± 0.4
Direct outcome	92.3 ± 0.1	76.8 ± 0.5	71.7 ± 0.6	61.9 ± 0.6	54.0 ± 0.7
Online verifier	92.1 ± 0.1	76.0 ± 0.4	70.7 ± 0.5	60.9 ± 0.5	52.8 ± 0.7
V-STaR-style (adapt.)	92.2 ± 0.1	76.4 ± 0.4	71.2 ± 0.5	61.4 ± 0.6	53.4 ± 0.7
KDCE	92.3 ± 0.2	76.8 ± 0.6	71.7 ± 0.7	61.9 ± 0.8	54.0 ± 0.9
Qwen3.8-27B					
Base	97.0 ± 0.0	90.0 ± 0.0	86.0 ± 0.0	79.0 ± 0.1	70.0 ± 0.1
Fixed verifier	97.2 ± 0.0	90.6 ± 0.1	86.8 ± 0.1	79.9 ± 0.1	71.2 ± 0.2
Success filtering	97.3 ± 0.0	91.1 ± 0.1	87.4 ± 0.2	80.6 ± 0.2	72.1 ± 0.3
Direct outcome	97.7 ± 0.1	92.6 ± 0.2	89.3 ± 0.3	82.8 ± 0.4	75.0 ± 0.5
Online verifier	97.5 ± 0.1	92.1 ± 0.2	88.8 ± 0.3	82.1 ± 0.3	74.1 ± 0.4
V-STaR-style (adapt.)	97.6 ± 0.1	92.3 ± 0.2	89.0 ± 0.3	82.4 ± 0.3	74.5 ± 0.5
KDCE	97.5 ± 0.1	92.0 ± 0.3	88.6 ± 0.4	82.0 ± 0.5	73.9 ± 0.6
Qwen3.6 MoE					
Base	95.0 ± 0.0	86.0 ± 0.0	79.0 ± 0.1	70.0 ± 0.1	62.0 ± 0.1
Fixed verifier	95.2 ± 0.0	86.8 ± 0.1	80.1 ± 0.2	71.2 ± 0.2	63.4 ± 0.2
Success filtering	95.4 ± 0.0	87.3 ± 0.2	80.8 ± 0.2	71.9 ± 0.3	64.4 ± 0.3
Direct outcome	95.9 ± 0.1	89.1 ± 0.3	83.2 ± 0.4	74.7 ± 0.5	67.7 ± 0.6
Online verifier	95.7 ± 0.1	88.6 ± 0.3	82.5 ± 0.4	73.9 ± 0.4	66.8 ± 0.5
V-STaR-style (adapt.)	95.8 ± 0.1	88.9 ± 0.3	82.9 ± 0.4	74.3 ± 0.4	67.2 ± 0.6
KDCE	95.8 ± 0.1	88.9 ± 0.4	82.9 ± 0.5	74.3 ± 0.6	67.3 ± 0.8

Method	MATH-500	AIME 25	AIME 26	HMMT 25	HMMT 26
Gemma 4 12B					
Base	91.0 ± 0.0	74.0 ± 0.1	68.0 ± 0.1	59.0 ± 0.1	50.0 ± 0.2
Fixed verifier	91.3 ± 0.1	75.1 ± 0.2	69.4 ± 0.2	60.4 ± 0.3	51.7 ± 0.3
Success filtering	91.5 ± 0.1	75.9 ± 0.2	70.3 ± 0.3	61.3 ± 0.3	52.8 ± 0.4
Direct outcome	92.3 ± 0.1	78.5 ± 0.5	73.5 ± 0.5	64.6 ± 0.6	56.7 ± 0.7
Online verifier	92.1 ± 0.1	77.8 ± 0.4	72.5 ± 0.5	63.7 ± 0.5	55.6 ± 0.6
V-STaR-style (adapt.)	92.2 ± 0.1	78.2 ± 0.4	73.0 ± 0.5	64.2 ± 0.6	56.2 ± 0.7
KDCE	92.3 ± 0.2	78.5 ± 0.6	73.4 ± 0.7	64.5 ± 0.8	56.6 ± 0.9
Gemma 4 MoE					
Base	94.0 ± 0.0	85.0 ± 0.1	79.0 ± 0.1	69.0 ± 0.1	60.0 ± 0.1
Fixed verifier	94.2 ± 0.0	85.8 ± 0.1	80.1 ± 0.2	70.2 ± 0.2	61.5 ± 0.3
Success filtering	94.4 ± 0.1	86.4 ± 0.2	80.8 ± 0.2	71.0 ± 0.3	62.5 ± 0.3
Direct outcome	95.0 ± 0.1	88.3 ± 0.3	83.2 ± 0.4	73.8 ± 0.5	65.9 ± 0.6
Online verifier	94.8 ± 0.1	87.7 ± 0.3	82.5 ± 0.4	73.0 ± 0.4	64.9 ± 0.5
V-STaR-style (adapt.)	94.9 ± 0.1	88.0 ± 0.3	82.9 ± 0.4	73.4 ± 0.5	65.4 ± 0.6
KDCE	94.9 ± 0.1	88.1 ± 0.4	83.0 ± 0.5	73.5 ± 0.6	65.5 ± 0.8
Olmo 3 7B					
Base	84.0 ± 0.1	47.0 ± 0.2	40.0 ± 0.2	29.0 ± 0.2	23.0 ± 0.3
Fixed verifier	84.4 ± 0.1	48.7 ± 0.3	42.0 ± 0.4	31.0 ± 0.4	25.2 ± 0.4
Success filtering	84.7 ± 0.1	49.9 ± 0.4	43.3 ± 0.5	32.3 ± 0.5	26.6 ± 0.6
Direct outcome	85.8 ± 0.2	54.0 ± 0.7	47.9 ± 0.8	36.8 ± 0.9	31.7 ± 1.0
Online verifier	85.5 ± 0.2	52.8 ± 0.7	46.6 ± 0.7	35.5 ± 0.8	30.3 ± 0.9
V-STaR-style (adapt.)	85.6 ± 0.2	53.4 ± 0.7	47.3 ± 0.8	36.2 ± 0.8	31.0 ± 0.9
KDCE	85.7 ± 0.3	53.6 ± 0.9	47.6 ± 1.1	36.4 ± 1.1	31.3 ± 1.2
Olmo 3.1 32B					
Base	93.0 ± 0.0	76.0 ± 0.1	68.0 ± 0.1	55.0 ± 0.2	46.0 ± 0.2
Fixed verifier	93.3 ± 0.0	77.1 ± 0.2	69.4 ± 0.2	56.5 ± 0.3	47.8 ± 0.3
Success filtering	93.5 ± 0.1	77.8 ± 0.2	70.3 ± 0.3	57.5 ± 0.4	48.9 ± 0.4
Direct outcome	94.1 ± 0.1	80.3 ± 0.4	73.5 ± 0.5	60.9 ± 0.6	53.0 ± 0.8
Online verifier	93.9 ± 0.1	79.6 ± 0.4	72.5 ± 0.5	60.0 ± 0.6	51.9 ± 0.7
V-STaR-style (adapt.)	94.0 ± 0.1	80.0 ± 0.4	73.0 ± 0.5	60.5 ± 0.6	52.5 ± 0.7
KDCE	94.1 ± 0.1	80.3 ± 0.6	73.4 ± 0.7	60.9 ± 0.8	53.0 ± 1.0

Published SOL-VER endpoints in Table 11 retain their source protocol (Lin et al., 2025, v2). The source uses greedy three-shot generation, MBPP rather than MBPP+, and a LiveCodeBench subset not established as identical to v6. The v2 numerical source and v4 method description concern the same work, not independent methods. Source relative-change entries are not substituted for endpoint values; differences in prompts, language, inference/search budgets, supervision, and checkers preclude treating these as matched controls.

Table 11: SOL-VER source-protocol endpoints (Lin et al., 2025, v2), separate from the controlled KDCE matrix. All rows use Llama 3.1 8B; source relative-change entries are not reused.

Source model	Benchmark	Source condition	Reported pass@1
Llama 3.1 8B	MBPP	Baseline	38.60
Llama 3.1 8B	MBPP	Iter3 + DPO	40.98
Llama 3.1 8B	LiveCodeBench	Baseline	18.23
Llama 3.1 8B	LiveCodeBench	Iter3 + DPO	27.24

C.2 Standalone inference and held-out structures

Table 12(a) evaluates the same checkpoint with standalone generation and sampled verifier-assisted inference. Standalone costs one normalized inference unit; Oracle@8 is finite-pool coverage, and Selected@8 includes all generation/scoring costs. Greedy pass@1 and sampled pass@k use their own locked decoding protocols and are not interchangeable. Panel (b) holds out parent templates, action combinations, and dependency lengths before data generation. These controlled-family probes do not alter the public benchmark aggregate: new combinations improve, but longer dependencies favor the online verifier.

Table 12: Deployment and generalization readouts. (a) Qwen3.5-9B code, with assisted-inference cost relative to a standalone unit. (b) Held-out controlled-family success (%). Neither finite coverage nor absent observed strategies establishes support expansion.

(a) Standalone versus selected inference

Policy	Standalone	Oracle@8	Selected@8	Inference cost
Base	66.0	84.0	76.0	8.6
KDCE	74.1	91.0	86.2	8.9

(b) Held-out structural probes

Held-out split	Base	Online verifier	KDCE
Held-out templates	51.0	57.8	60.1
New action combinations	38.0	43.4	46.2
Longer dependency chains	21.0	23.0	22.7

C.3 Resource ledger and balanced aggregation

Environment and learning-compute caps are separate normalized coordinates, not FLOPs or dollar budgets shared across architectures. Acquisition, failures, paired edits, extra checking, initialization, and calibration consume the environment cap; generation, all ensemble scoring, edits, optimization, and development consume compute. Audits and locked benchmark inference have separate ledgers and never supply training feedback. One trained model/domain evaluated on several tests incurs one training cost. Deployment amortization is outside the reported estimand.

Table 13: Compute-cap fractions. (a) Qwen3.5-9B/code components: generation, scoring, edit generation, solver/verifier training, development, and initialization. Totals precede rounding. (b) Each model row applies identically to code, mathematics, and interaction. Trained methods use environment fraction 1.00 except fixed verifier (0.95); Base uses no added training resources.

(a) Component allocation

Method	Gen.	Score	Edit	S train	V train	Dev.	Init.	Total
Fixed verifier	0.325	0.106	0.000	0.265	0.000	0.038	0.023	0.756
Success filtering	0.388	0.000	0.000	0.282	0.000	0.021	0.014	0.706
Direct outcome	0.440	0.000	0.000	0.364	0.000	0.025	0.017	0.847
Online verifier	0.302	0.115	0.000	0.257	0.151	0.035	0.027	0.887
V-STaR-style (adapt.)	0.300	0.150	0.000	0.253	0.169	0.037	0.028	0.937
KDCE	0.277	0.128	0.079	0.267	0.168	0.040	0.030	0.988

(b) Complete model-level allocation

Model	Fixed	Filter	Direct	Online	V-STaR (adapt.)	KDCE
Qwen3.5-4B	0.750	0.700	0.840	0.880	0.930	0.980
Qwen3.5-9B	0.756	0.706	0.847	0.887	0.937	0.988
Qwen3.8-27B	0.744	0.694	0.833	0.873	0.923	0.972
Qwen3.6 MoE	0.753	0.703	0.843	0.884	0.934	0.984
Gemma 4 12B	0.750	0.700	0.840	0.880	0.930	0.980
Gemma 4 MoE	0.747	0.697	0.837	0.876	0.926	0.976
Olmo 3 7B	0.750	0.700	0.840	0.880	0.930	0.980
Olmo 3.1 32B	0.753	0.703	0.843	0.884	0.934	0.984

Table 13 reports component and model-level allocations. Hardware, precision, memory, tokens, latency, and parallelism are separate attributes; normalized efficiency does not establish speed or hardware-cost savings. A matched-useful-compute control may reallocate saved resources under a fixed development rule.

Within each domain, average tests in each benchmark family first (EvalPlus, AIME years, and HMMT years are respective families), then weight families equally. Average models within Qwen/Gemma/Olmo, then families and domains equally. All 24 settings, including negative results, enter Table 14. Compute each paired difference within the same configuration. The gain-per-used-

compute ratio is

$$\frac{(D_{\mathrm{KDCE}} - D_0)/c_{\mathrm{KDCE}}}{(D_{\mathrm{ref}} - D_0)/c_{\mathrm{ref}}},$$

with a family-balanced geometric mean. Reference gains are positive here; otherwise report a difference, not an undefined ratio or a dropped setting. Ratios above one are counted before rounding.

Table 14: All model/domain effects and gain-per-used-compute ratios. The prespecified reference is the online verifier for code and interactive settings and direct outcome for mathematics. Domain scores use family-aware task means; brackets show paired configuration ranges. Positive-setting counts use unrounded paired differences; negative and inefficient settings are retained.

Model	Domain	Base	Ref.	KDCE	Δ pp [range]	Eff.
Qwen3.5-4B	Code	46.17	50.81	52.20	1.4 [1.2, 1.6]	1.17
Qwen3.5-4B	Math	53.67	59.32	59.41	0.1 [-0.1, 0.3]	0.87
Qwen3.5-4B	Interactive	10.00	17.97	17.65	-0.3 [-0.7, 0.1]	0.86
Qwen3.5-9B	Code	55.33	59.24	60.37	1.1 [1.0, 1.3]	1.16
Qwen3.5-9B	Math	70.50	74.80	74.83	0.0 [-0.1, 0.2]	0.86
Qwen3.5-9B	Interactive	18.50	26.00	28.10	2.1 [1.7, 2.5]	1.15
Qwen3.8-27B	Code	70.00	72.65	72.96	0.3 [0.2, 0.4]	1.00
Qwen3.8-27B	Math	86.50	89.15	88.60	-0.6 [-0.6, -0.5]	0.68
Qwen3.8-27B	Interactive	38.00	44.34	46.37	2.0 [1.7, 2.3]	1.19
Qwen3.6 MoE	Code	64.33	67.52	68.05	0.5 [0.4, 0.7]	1.05
Qwen3.6 MoE	Math	81.17	84.42	84.18	-0.2 [-0.3, -0.2]	0.79
Qwen3.6 MoE	Interactive	30.00	36.83	39.36	2.5 [2.2, 2.9]	1.23
Gemma 4 12B	Code	57.33	61.04	62.00	1.0 [0.8, 1.1]	1.13
Gemma 4 12B	Math	72.17	76.32	76.25	-0.1 [-0.2, 0.0]	0.84
Gemma 4 12B	Interactive	19.50	26.95	28.88	1.9 [1.6, 2.3]	1.13
Gemma 4 MoE	Code	62.50	65.78	66.43	0.7 [0.5, 0.8]	1.08
Gemma 4 MoE	Math	80.17	83.52	83.33	-0.2 [-0.3, -0.1]	0.81
Gemma 4 MoE	Interactive	28.50	35.43	37.09	1.7 [1.3, 2.0]	1.11
Olmo 3 7B	Code	38.00	43.29	44.79	1.5 [1.3, 1.7]	1.15
Olmo 3 7B	Math	51.17	57.01	56.72	-0.3 [-0.5, -0.1]	0.81
Olmo 3 7B	Interactive	4.00	12.29	11.71	-0.6 [-1.0, -0.2]	0.84
Olmo 3.1 32B	Code	54.67	58.61	59.56	1.0 [0.8, 1.1]	1.11
Olmo 3.1 32B	Math	71.83	75.99	75.96	0.0 [-0.2, 0.1]	0.85
Olmo 3.1 32B	Interactive	15.50	23.17	25.24	2.1 [1.7, 2.5]	1.14

Performance targets are fixed increments above initial domain scores, chosen on development before final audits. The first target-reaching checkpoint is retrospective and never alters training duration. Table 15 retains 14 KDCE and 16 reference nonattainers as right-censored within the cap; no unattained target receives extrapolated or zero cost. Source collection, reusable feedback, and any declared amortization must be included before interpreting target-call savings.

Table 15: Offline target-reaching audit. Values beyond round 8 or cost 1.00 denote right-censoring, not extrapolated measurements. Targets are preset increments, not test-tuned stopping levels.

Model	Domain	Target	Ref. round	KD round	Ref. cost	KD cost
Qwen3.5-4B	Code	52.17	> 8	8	> 1.00	0.98
Qwen3.5-4B	Math	59.67	> 8	> 8	> 1.00	> 1.00
Qwen3.5-4B	Interactive	14.00	2	3	0.22	0.37
Qwen3.5-9B	Code	61.33	> 8	> 8	> 1.00	> 1.00
Qwen3.5-9B	Math	76.50	> 8	> 8	> 1.00	> 1.00
Qwen3.5-9B	Interactive	22.50	2	2	0.22	0.25
Qwen3.8-27B	Code	76.00	> 8	> 8	> 1.00	> 1.00
Qwen3.8-27B	Math	92.50	> 8	> 8	> 1.00	> 1.00
Qwen3.8-27B	Interactive	42.00	3	3	0.33	0.36
Qwen3.6 MoE	Code	70.33	> 8	> 8	> 1.00	> 1.00
Qwen3.6 MoE	Math	87.17	> 8	> 8	> 1.00	> 1.00

Model	Domain	Target	Ref. round	KD round	Ref. cost	KD cost
Qwen3.6 MoE	Interactive	34.00	2	2	0.22	0.25
Gemma 4 12B	Code	63.33	> 8	> 8	> 1.00	> 1.00
Gemma 4 12B	Math	78.17	> 8	> 8	> 1.00	> 1.00
Gemma 4 12B	Interactive	23.50	2	2	0.22	0.24
Gemma 4 MoE	Code	68.50	> 8	> 8	> 1.00	> 1.00
Gemma 4 MoE	Math	86.17	> 8	> 8	> 1.00	> 1.00
Gemma 4 MoE	Interactive	32.50	2	2	0.22	0.24
Olmo 3 7B	Code	44.00	> 8	6	> 1.00	0.73
Olmo 3 7B	Math	57.17	> 8	> 8	> 1.00	> 1.00
Olmo 3 7B	Interactive	8.00	2	3	0.22	0.37
Olmo 3.1 32B	Code	60.67	> 8	> 8	> 1.00	> 1.00
Olmo 3.1 32B	Math	77.83	> 8	> 8	> 1.00	> 1.00
Olmo 3.1 32B	Interactive	19.50	2	2	0.22	0.25

C.4 Common-schedule checkpoints and complete curves

Table 16 places rounds 2, 4, 6, and 8 side by side for every model/domain. The reference is the online verifier for code/interaction and the direct-outcome learner for mathematics. Table 17 lists their corresponding used compute fractions; both branches use environment fractions 0.250, 0.500, 0.750, and 1.000 at these rounds. These are prefixes of one common schedule, not separately retuned budget optima. Figures 7 and 8 retain all 24 model/domain trajectories. The gallery reuses the endpoint runs and is not independent corroboration.

Table 16: All schedule prefixes: each cell is reference / KDCE domain success (%). Shared resources are factorized into Table 17; The matrix covers every model/domain/checkpoint combination.

Model	Domain	Round 2	Round 4	Round 6	Round 8
Qwen3.5-4B	Code	48.9 / 49.0	50.1 / 50.7	50.6 / 51.6	50.8 / 52.2
Qwen3.5-4B	Math	57.0 / 56.4	58.4 / 58.0	59.1 / 58.9	59.3 / 59.4
Qwen3.5-4B	Interactive	14.7 / 13.6	16.7 / 15.7	17.6 / 16.9	18.0 / 17.7
Qwen3.5-9B	Code	57.6 / 57.7	58.6 / 59.1	59.1 / 59.9	59.2 / 60.4
Qwen3.5-9B	Math	73.0 / 72.5	74.1 / 73.7	74.6 / 74.4	74.8 / 74.8
Qwen3.5-9B	Interactive	22.9 / 23.0	24.8 / 25.7	25.6 / 27.2	26.0 / 28.1
Qwen3.8-27B	Code	71.6 / 71.4	72.2 / 72.2	72.5 / 72.7	72.7 / 73.0
Qwen3.8-27B	Math	88.1 / 87.5	88.7 / 88.1	89.0 / 88.4	89.2 / 88.6
Qwen3.8-27B	Interactive	41.7 / 41.9	43.3 / 44.2	44.0 / 45.6	44.3 / 46.4
Qwen3.6 MoE	Code	66.2 / 66.1	67.0 / 67.1	67.4 / 67.7	67.5 / 68.0
Qwen3.6 MoE	Math	83.1 / 82.6	83.9 / 83.4	84.3 / 83.9	84.4 / 84.2
Qwen3.6 MoE	Interactive	34.0 / 34.4	35.8 / 37.0	36.5 / 38.5	36.8 / 39.4
Gemma 4 12B	Code	59.5 / 59.5	60.5 / 60.8	60.9 / 61.6	61.0 / 62.0
Gemma 4 12B	Math	74.6 / 74.1	75.7 / 75.2	76.1 / 75.9	76.3 / 76.2
Gemma 4 12B	Interactive	23.9 / 23.9	25.8 / 26.5	26.6 / 28.0	26.9 / 28.9
Gemma 4 MoE	Code	64.4 / 64.4	65.3 / 65.4	65.6 / 66.1	65.8 / 66.4
Gemma 4 MoE	Math	82.1 / 81.7	83.0 / 82.5	83.4 / 83.0	83.5 / 83.3
Gemma 4 MoE	Interactive	32.6 / 32.6	34.3 / 34.9	35.1 / 36.3	35.4 / 37.1
Olmo 3 7B	Code	41.1 / 41.2	42.5 / 43.1	43.0 / 44.2	43.3 / 44.8
Olmo 3 7B	Math	54.6 / 53.8	56.1 / 55.3	56.7 / 56.2	57.0 / 56.7
Olmo 3 7B	Interactive	8.9 / 7.6	11.0 / 9.8	11.9 / 11.0	12.3 / 11.7
Olmo 3.1 32B	Code	57.0 / 57.0	58.0 / 58.3	58.4 / 59.1	58.6 / 59.6
Olmo 3.1 32B	Math	74.3 / 73.8	75.3 / 74.9	75.8 / 75.6	76.0 / 76.0
Olmo 3.1 32B	Interactive	20.0 / 20.1	22.0 / 22.8	22.8 / 24.3	23.2 / 25.2

Table 17: Used compute at each prefix. Each cell is online verifier / direct outcome / KDCE; online is the reference for code and interaction, direct for mathematics. The shared environment fractions are given in the text.

Model	Round 2	Round 4	Round 6	Round 8
Qwen3.5-4B	0.220 / 0.210 / 0.245	0.440 / 0.420 / 0.490	0.660 / 0.630 / 0.735	0.880 / 0.840 / 0.980
Qwen3.5-9B	0.222 / 0.212 / 0.247	0.444 / 0.423 / 0.494	0.665 / 0.635 / 0.741	0.887 / 0.847 / 0.988
Qwen3.8-27B	0.218 / 0.208 / 0.243	0.436 / 0.417 / 0.486	0.655 / 0.625 / 0.729	0.873 / 0.833 / 0.972
Qwen3.6 MoE	0.221 / 0.211 / 0.246	0.442 / 0.422 / 0.492	0.663 / 0.633 / 0.738	0.884 / 0.843 / 0.984
Gemma 4 12B	0.220 / 0.210 / 0.245	0.440 / 0.420 / 0.490	0.660 / 0.630 / 0.735	0.880 / 0.840 / 0.980
Gemma 4 MoE	0.219 / 0.209 / 0.244	0.438 / 0.418 / 0.488	0.657 / 0.627 / 0.732	0.876 / 0.837 / 0.976
Olmo 3 7B	0.220 / 0.210 / 0.245	0.440 / 0.420 / 0.490	0.660 / 0.630 / 0.735	0.880 / 0.840 / 0.980
Olmo 3.1 32B	0.221 / 0.211 / 0.246	0.442 / 0.422 / 0.492	0.663 / 0.633 / 0.738	0.884 / 0.843 / 0.984

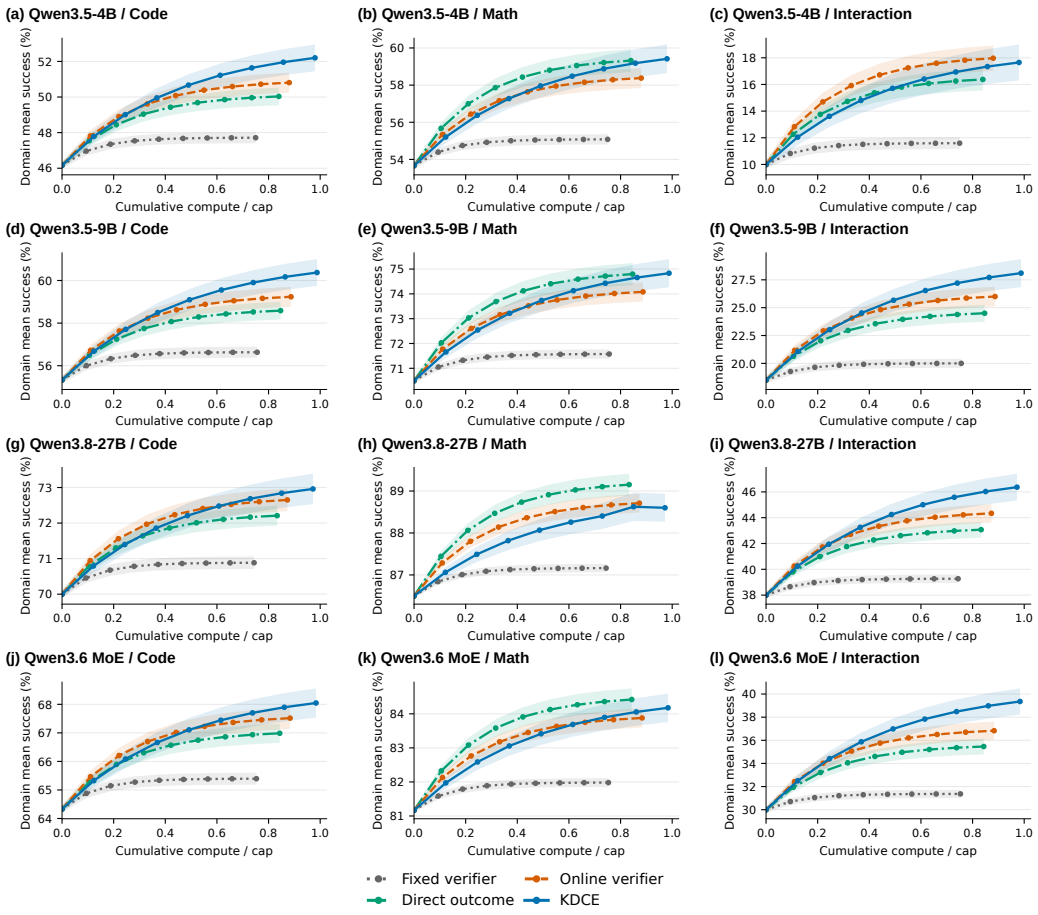

Figure 7: Complete Qwen learning curves. Rows: Qwen3.5-4B, Qwen3.5-9B, Qwen3.8-27B, Qwen3.6 MoE. Columns: code, mathematics, interaction. Panels share method colors and a single legend. Shading spans the five paired configurations; curves are unsmoothed. Axes show domain success and used compute-cap fraction, not absolute FLOPs.

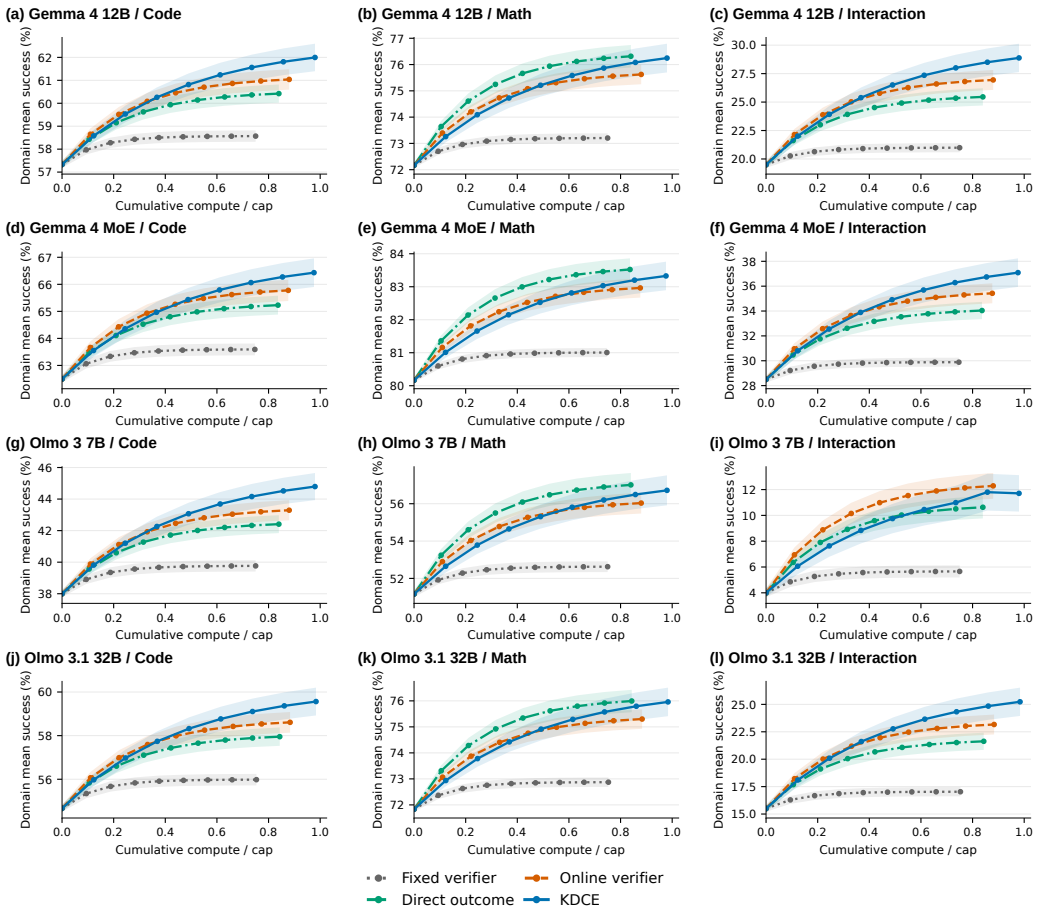

Figure 8: Complete Gemma/Olmo learning curves. Rows: Gemma 4 12B, Gemma 4 MoE, Olmo 3 7B, Olmo 3.1 32B. Columns: code, mathematics, interaction. Panels share method colors and a single legend. Shading spans the five paired configurations; curves are unsmoothed. Axes show domain success and used compute-cap fraction, not absolute FLOPs.

D Directional mechanisms and matched controls

D.1 Renewed judgment: shared evidence and pathway interventions

The plateau rule uses development improvement within 0.4 pp over three checkpoints; a nonsignificant slope alone is insufficient. Restart branches share checkpoints, optimizer state, candidate source, and initial evidence under equal continuation caps. This origin is distinct from the broad matrix endpoint. Online continuation may collect different trajectories and charges them.

In the identical-log experiment, every learner receives the same task/candidate records, queried outcomes, local fields, executed edit pairs, and costs, with no additional queries or reference answers. Success SFT uses observed positives; outcome DPO uses observed contrasts; trace-to-policy may exploit recorded feedback and successful edits but not invent repair labels. Online and KDCE verifiers share fields and the downstream policy updater. Table 18(a) shows 3.8 pp for KDCE versus 3.2 for online verification and 3.6 for trace-to-policy. Native on-policy direct-outcome learning remains a separate online comparator, not its fixed-log substitute.

The phase audit freezes policy during verifier learning and verifier during policy learning. Fixed-reference K_{ref} and current-bank C, R, K, G have different distributions and units, as labeled in panel (b). Blocking refreshed selection retains verifier training and compute, common candidate banks, successful replay, target quotas, and policy-update weights. Panel (c)'s rescue branches both start from the blocked learner at R; comparing restoration to another history would confound the effect. Reference-pool gains alone do not demonstrate improved policy training.

Withdrawing new evidence retains replay and the same non-environment compute allowance, while saving execution cost. Old evidence supports another 1.8 pp before flattening (Figure 9f). This measures remaining uptake, not information creation or guaranteed recovery after new feedback resumes.

Table 18: Forward-path controls. (a) Identical-log changes (pp); direct learners have no trained verifier. (b) Fixed-reference accuracy and standalone D use percent; current-bank C, R, K, G use probabilities. (c) S and R are separate matched starts; gains follow a policy update, not instantaneous scorer replacement.

(a) Different uses of identical execution evidence

Learner	ΔD (pp)	ΔK_{ref} (pp)
Success SFT	2.0	0.0
Outcome DPO	2.5	0.0
Trace-to-policy	3.6	0.0
Online verifier	3.2	4.7
KDCE	3.8	5.1

(b) Measurements before and after each learning phase

Phase	D (%)	K_{ref} (%)	Brier	NLL	C	R	K	G
Before	69.000	74.000	0.194	0.570	0.900	0.840	0.756	0.066
Verifier updated	69.000	79.100	0.164	0.490	0.900	0.890	0.801	0.111
Policy updated	71.700	79.100	0.164	0.490	0.915	0.900	0.824	0.107

(c) Blocked and restored selection paths

Branch	Origin	Start	Future ΔD	Compute
Frozen	S	69.0	0.9	0.7
Blocked	S	69.0	1.0	1.0
Enabled	S	69.0	2.7	1.0
Stay cut	R	70.0	0.5	1.0
Restore	R	70.0	1.9	1.0

D.2 Capacity, acquisition, and update schedule

Capacity controls match adapters, ensemble size, fields, and optimizer caps. Matching count, length, and development-assessed difficulty supplements the total-effect comparison; it must not remove correct-target quality, the proposed mediator. Calibration-only preserves old ranking; blocked-output training exposes no refreshed selection to the policy, checking unintended parameter coupling under a frozen backbone.

The query-by-learner factorial crosses random/active acquisition with direct/KDCE use. Within a query rule, learners share the exact log; between rules, the interface and resource cap are shared but logs may differ. Schedule variants share records and total update allowances, recording the verifier version behind each policy-data snapshot. Fresh selection helps on changing distributions but little on stationary ones. More frequent scoring/calibration is charged, so verifier-first ordering is an operating choice, not a universal principle.

D.3 Policy-supplied experience and conditional knowledge

Experience-source branches clone the same receiving verifier and match task distribution, query rule, execution cap, decoding, and update budget. Old, new, mixed, and matched-new evidence is evaluated on an independent common reference pool. The secondary matching balances length, difficulty, and success fraction, but not state novelty, which is part of the proposed mechanism. Qwen/Gemma code/tool settings are separate replications; the Qwen code contrast is small. In long-run fixed-source comparisons, only the evidence-generating adapter is frozen: receiving policies still update and generate their own selection banks.

Failure-only additions share confirmed policy successes, historical verifier replay, and the downstream bank. New failures enter verifier outcome/local-field losses only, never solver prompts or

Table 19: Policy gains (pp) under alternative-explanation controls. Shared information and capacities are preserved; matched presentation statistics are supplementary to the total selection effect.

(a) Capacity and target-presentation controls

Controlled comparison	ΔD
KDCE, common evidence	3.8
Online verifier, same ensemble and fields	3.2
KDCE, matched count/length/difficulty	3.6
Online, matched count/length/difficulty	3.1
Calibrate only, preserve old ranking	1.1
Train verifier, block all selection output	1.0

(b) Acquisition strategy by learner

Query	Learner	ΔD
Random	Direct	2.3
Active	Direct	2.5
Random	KDCE	3.5
Active	KDCE	3.8

(c) Update timing on changing and stationary distributions

Schedule	Changing distribution	Stationary distribution
Current verifier first	3.8	2.1
Stale snapshot	2.2	2.0
Synchronous old snapshots	2.3	2.1
Renew every other round	2.8	2.1

imitation targets. Tied terminal failures create no fabricated ranking preferences. At zero successful coverage, judgment can improve but no correct policy target becomes available; R is then undefined. Oracle insertion remains a separate diagnostic.

Content probes change one action at fixed state or one visible precondition at fixed action, executing both variants under matched resets. All parent variants stay in one partition; labels describe the tested edit, not a uniquely inferred internal cause. No-local reallocates execution cost; shuffling preserves terminal labels and format/length while breaking local correspondences; placebo inserts non-informative tokens. These are distinct controls. The weak long-horizon gain limits generalization, and no separate explanation network, rule store, or oracle router is credited.

E Reliability, ablations, and operating boundaries

E.1 Candidate banks, selective risk, and independent checks

The ablations in Table 2 share the Qwen3.5-9B LCB endpoint. Freezing stops verifier learning; stale selection still trains it but supplies old judgments to the policy. No-local reallocates saved calls and keeps returned fields; random querying changes only acquisition; no-ranking retains all outcome/local losses. Raw argmax accepts one unexecuted candidate per task with its provenance intact. The small no-ranking effect does not establish indispensability.

Nested pools guarantee nondecreasing coverage, not monotone selected accuracy (Table 21). The acceptance audit chooses at most one target per original task. At acceptance a and error e, correct yield is $a(1-e)$, not accepted accuracy or total retained trajectories. At zero acceptance, error is undefined and yield is zero. Higher correct yield can coexist with more contamination; the subsequent policy update, not risk alone, resolves the trade-off. Every N requires its own calibrated pipeline, and extra queries are charged.

Independent checks re-execute the same outputs under withheld tests; added stress checks can only remove successes. The primary LCB values are the independent-check endpoints, not visible-test rates or independent corroborating runs. Sealed audits inspect high-score failures, parser exploits, timeouts, and incomplete-checker successes. Training-event calibration does not certify a stronger checker or a shifted deployment condition. Figure 9(a–d) and the table show these distinctions.

Table 20: Reverse-path and content controls. (a) Receiving-verifier ΔK_{ref} (pp), with matched-new balancing observed composition. (b) Identical confirmed policy successes; gains are pp and coverage is a probability. (c) Outcome accuracy (%) on held-out probes.

(a) Old versus updated experience sources

Source setting	Old	New	Mixture	Matched new
Qwen / tool	2.6	5.3	4.8	4.5
Gemma / tool	2.3	4.6	4.2	3.8
Qwen / code	3.1	3.3	3.4	3.2
Gemma / code	2.6	4.0	3.7	3.5

(b) New failures supplied only to verification

Verifier-only addition	ΔD	ΔK_{ref}	Coverage
Old failures	1.4	2.2	0.8
New failures	3.2	4.9	0.8
Repeated failures	1.5	2.3	0.8
No coverage	0.0	3.0	0.0

(c) Unseen actions and applicability conditions

Probe	KDCE	No local	Shuffled	Placebo
Unseen action	78.0	72.0	67.0	72.2
Changed condition	73.0	62.0	59.0	62.3
Long horizon	58.0	57.0	54.0	57.1

Table 21: Reliability readouts. (a) Nested-bank entries are probabilities; extra queries are cap fractions. Forced-choice K is separate from abstaining acceptance/risk/yield. (b) Errors at matched acceptance are percentages. (c) Nested checker success (%) from the same policy endpoints.

(a) Candidate count, selection, and useful yield

N	C	K old	K new	Accept	Error	Correct yield	Extra queries
1	0.660	0.660	0.660	0.300	0.010	0.297	0.000
8	0.840	0.760	0.780	0.620	0.023	0.606	0.020
32	0.900	0.780	0.827	0.720	0.030	0.698	0.040
128	0.940	0.730	0.846	0.700	0.050	0.665	0.090

(b) Risk at matched acceptance

Acceptance	KDCE error (%)	Online error (%)	Shift error (%)	Extra query fraction
0.150	0.300	0.700	2.000	0.020
0.300	0.700	1.500	5.000	0.025
0.500	1.500	3.800	9.000	0.030
0.720	3.000	6.700	14.000	0.040
0.900	8.000	12.000	21.000	0.070
1.000	15.400	18.000	29.000	0.100

(c) Visible, independent, and added stress checks

Method	Visible check	Independent check	Added stress check
Direct outcome	75.7	71.2	68.1
Online verifier	76.8	72.3	69.6
KDCE	78.6	74.1	71.2

E.2 FEEDBACK QUALITY, SHIFT, AND NON-CONVERSION

Table 22 and Figure 9(e–h) distinguish feedback dose/noise, withdrawal, rule shift, and non-conversion. Injected noise changes training feedback only; evaluation truth remains fixed. Random corruption, systematic checker error, missing fields, and stochastic outcomes are different perturbations, with replications charged. Combined-source ablations may be non-additive; record novelty and downstream selection changes, not just evidence counts.

Rule-shift experiments retain observable version/state information and track both old-rule retention and new-rule adaptation. Replay reduces but does not eliminate forgetting; unlabeled changes to the task objective cannot be treated as one fixed-utility problem. Withdrawal retains stored evidence and permitted computation (Appendix D.1).

The non-conversion settings are separate scenarios, not prevalence estimates. Better K_{ref} need not supply successful candidates, affect selected decisions, survive over-rejection, or be absorbed by the policy; systematic feedback error can damage both. Subtracting fixed-reference gain from policy gain does not give online G. Same-protocol identities $K = CR$, $G = K - D$, and $G_{t+1} - G_t = \Delta K_t - \Delta D_t$ are bookkeeping constraints. The results establish neither a universal feedback threshold nor unbounded improvement on bounded success metrics.

Table 22: Finite-window boundary results. (a) Late policy gain (pp) at each feedback/noise combination. (b) Old/new rule success (%) under distinct conditions. (c) Independent non-conversion scenarios; both gain columns are pp but refer to different evaluation protocols.

(a) Feedback dose and injected corruption

Feedback / cap	Noise 0.0	Noise 0.1	Noise 0.3
0.1	0.1	0.0	-0.5
0.3	1.0	0.5	-0.7
1.0	2.8	1.6	-1.4

(b) Adaptation and retention

Method	Old before	Old after	New after
KDCE + replay	71.0	69.4	64.1
No replay	71.0	63.0	65.0
Frozen verifier	71.0	70.1	55.2

(c) Prediction improvements without matching policy gains

Condition	ΔK_{ref}	ΔD
Useful local evidence	5.3	3.2
Zero successful coverage	3.0	0.0
Low policy absorption	4.8	0.2
Decision-irrelevant outcomes	4.5	0.1
Over-strict acceptance	5.1	0.3
Systematic feedback error	-1.8	-1.4

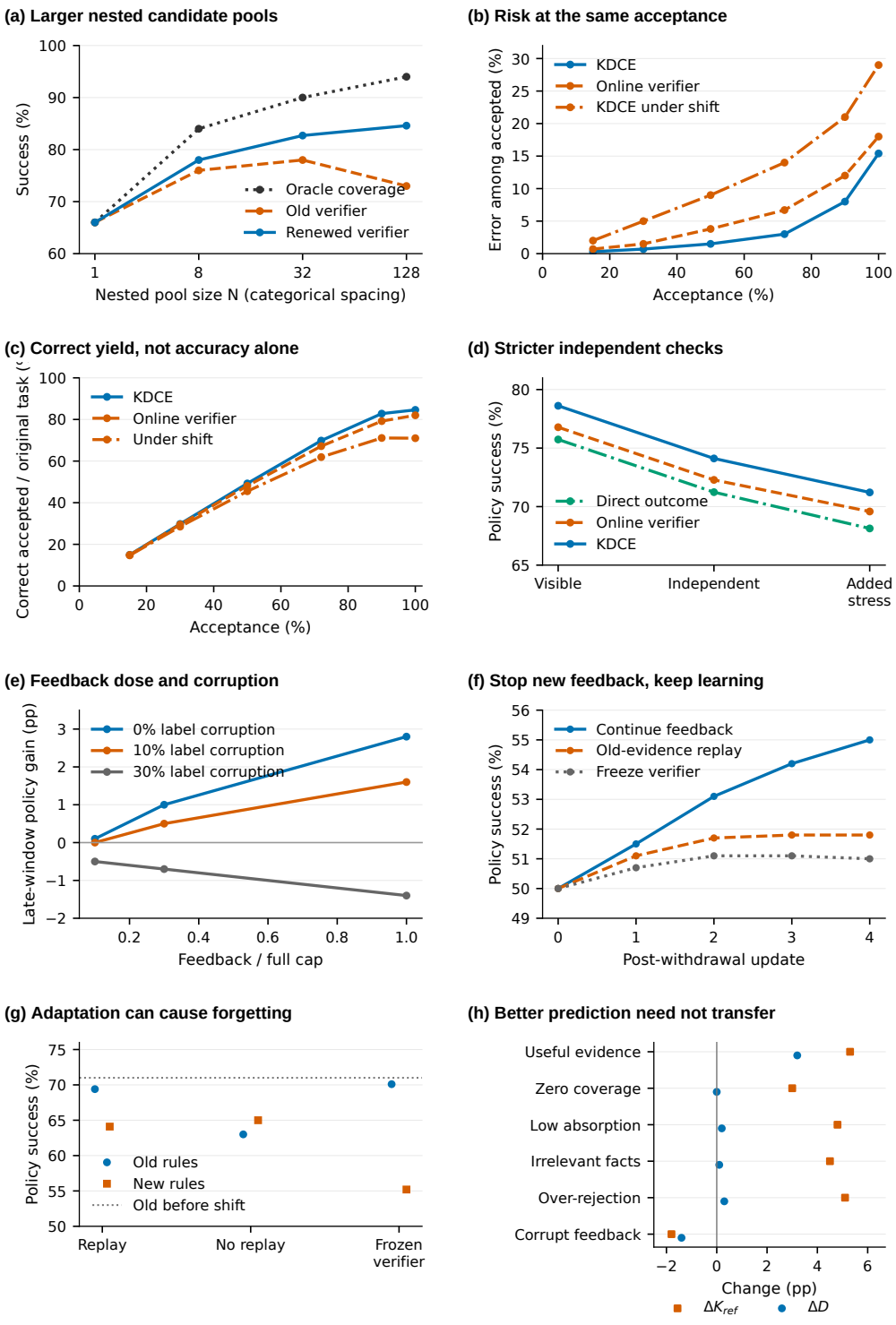

Figure 9: Reliability and operating boundaries. (a) Nested-pool forced selection; (b) risk at common acceptance, including shift; (c) correct targets per original task; (d) nested strict checks; (e) late gain under feedback dose/corruption; (f) continued feedback versus replay or frozen verification; (g) rule adaptation and retention; (h) distinct non-conversion scenarios. Panel-specific units and scales apply; bands and error bars show configuration sensitivity. Outcome axes in (a,d,f,g) are truncated. These are finite-window comparisons, not universal thresholds or task-sampling confidence bands.

F KDCE IMPLEMENTATION

F.1 ADAPTERS, REPLAY, AND NUMERICAL CONFIGURATION

Algorithm 1 Knowing–Doing Co-Evolution (full round-wise procedure)

Require: Tasks, environment $\mathcal{E}$, frozen backbone W_0, adapters θ_0, ϕ_0, rounds $T_{\max}$, budgets $\{B_t\}$, disjoint calibration splits

Initialize verifier and replay $\mathcal{M}_0$ with random queries of π_{θ_0} candidates, charged to B_0
for $t = 0, \ldots, T_{\max} - 1$ **do**
    Freeze W_0; reserve acquisition, edit, calibration, and extra-check budgets from B_t
    Freeze θ_t; sample $\mathcal{C}_t$ with π_{θ_t} and record provenance
    Query by ϕ_t disagreement/cost plus random coverage; collect execution outcomes
    Propose and execute local edits with frozen θ_t; add records to $\mathcal{E}_t$
    Train ϕ_{t+1} on $\mathcal{E}_t$ and replay using (4)
    Calibrate temperature and acceptance on separate tasks; rescore $\mathcal{C}_t$
    Execute reserved high-score, high-uncertainty candidates and form $\mathcal{D}_t$
    Freeze calibrated ϕ_{t+1}; update θ_t using (6), or copy it if $\mathcal{D}_t = \varnothing$
    Set result to θ_{t+1}; store training executions in $\mathcal{M}_{t+1}$
    Keep calibration/audit outcomes outside replay; finalize the cost ledger
end for
return $\pi_{\theta_{T_{\max}}}$ and $q_{\phi_{T_{\max}}}$

A frozen backbone supports one policy adapter and M independently initialized verifier adapters/output heads, using LoRA (Hu et al., 2022) and task-level bootstrap resampling (Lakshminarayanan et al., 2017). Shared-backbone errors can correlate; disagreement is evaluated empirically. Random initial-policy queries and verifier warm-start fitting are charged to B_0. Before valid calibration, policy training uses confirmed successes only. The policy is frozen during acquisition, edits, and verifier learning; verifiers and selector are frozen during policy learning. A shared trainable backbone is a separate architecture condition.

Replay stores actual executions by task, version, and outcome, preserving linked local pairs. Policy replay contains only compatible verified successes. Old-version records may train condition-specific verifier outputs but do not certify current policy positives. Each member predicts success probability $p_m = \sigma(z_m)$ and available local fields:

$$
\begin{aligned}
q_{\phi^{(m)}}(o \mid x, \tau) &= \prod_{\ell=0}^{L} q_{\phi^{(m)},\ell}(o_\ell \mid x, \tau), \\
-\log q_{\phi^{(m)}}(o_{\text{obs}} \mid x, \tau) &= -\sum_{\ell \in \mathcal{O}(e)} \log q_{\phi^{(m)},\ell}(o_\ell \mid x, \tau).
\end{aligned}
\tag{11}
$$

Here o_0 is final outcome and $\mathcal{O}(e)$ contains genuinely observed fields. Local heads mask future observations; missing fields marginalize out and are never imputed from verifier predictions. This factorization is a modeling choice, not physical independence. The ensemble is $q_\phi = M^{-1} \sum_m q_{\phi^{(m)}}$; for binary utility only the success head determines the score. Each member computes its own ranking loss using same-task, compatible-condition pairs with observed strict utility differences. Ties supply no preferences; stochastic comparisons require charged replications. Without pairs or local labels, their respective terms are omitted.

Policy likelihood covers generated solution/action tokens only, with environment responses masked. Successful feedback-conditioned edits are distilled as complete solutions under the original deployment context, not a failure prompt unavailable at standalone evaluation. Truncation and terminal handling are matched. Table 23 retains numerical settings; development freezes model/tokenizer revisions, fields, decoding, budgets, replay, calibration grids, and stopping rules before comparisons.

F.2 BUDGETED QUERIES AND CONTROLLED LOCAL EXECUTION

For binary outcomes, entropy disagreement provides the acquisition heuristic:

Table 23: Training, inference, and calibration configuration. Read each parameter–value column pair independently.

Parameter	Value	Parameter	Value
solver_rank	32	epochs	1
verifier_rank	32	train_temperature	0.8
lora_alpha	64	train_top_p	0.95
members	3	eval_temperature	0
N	32	generation_limit	16384
random_query_fraction	0.25	interaction_generated_tokens	32768
local_budget_fraction	0.23	interaction_tool_calls	100
rank_alpha	0.2	interaction_wall_hours	8
ranking_temperature	1	risk_target	0.05
lambda	1	min_accepted_calibration	100
omega	0.5	optimizer	AdamW
KL_eta	0.02	precision	bfloat16
replay_fraction	0.25	exact_prompt_bytes	24576 maximum
solver_lr	2e-05	adapter_target_modules	`q/k/v/o_proj`
verifier_lr	1e-05	batch_tokens	131072 per update
weight_decay	0.01	n_train_tasks	24000 per domain
gradient_clip	1		

$$b_\phi(x,\tau) = h(\bar{p}) - \frac{1}{M}\sum_{m=1}^{M} h(p_m),$$
$$h(p) = -p\log p - (1-p)\log(1-p), \qquad \text{priority}(x,\tau) = \frac{b_\phi(x,\tau)}{c(x,\tau)}. \tag{12}$$

Use $0\log 0 = 0$, and categorical entropy for multiclass outcomes. Predictive-information acquisition motivates the score (Houlsby et al., 2011), but the finite adapter ensemble is not a Bayesian posterior. Estimated positive cost is available before execution; the ledger records actual cost with conservative reservation. Apply task quotas and syntax-preserving duplicate filters, reserve a positive random-query share, then rank remaining candidates by priority. Fixed ties and budget-exhaustion rules apply. This explores the available proposal pool, not unseen support.

For queried failures or prediction errors, the frozen policy proposes one bounded local edit using actual feedback. Preserve its location, preconditions, and parent ID; reset tools or reuse the same tests/configuration and controllable randomness. A compatible stored original execution may serve as the first arm; otherwise rerun and charge it. Keep both outcomes whether the edit helps. Only supported utility differences make ranking pairs; successful full edits can become policy positives, failed edits remain verifier evidence. Changed preconditions require actual variant executions and stay in the parent's split. Terminal success differences identify the complete tested edit, not a unique internal cause; localization requires returned evidence, and unexecuted explanations remain hypotheses.

Each round reserves acquisition, edit, calibration, and extra-check resources with a fixed transfer rule. For paid events $\mathcal{Q}_t$,

$$\sum_{q\in\mathcal{Q}_t} c(q) \le B_t. \tag{13}$$

Repeated executions are separate events; reusing a record is not a new call. Multi-field returns are charged once. Extra checks remain paid this round even when used for verifier training only next round. Budget accounting and cross-learner controls follow Appendices A.4 and C.3.

F.3 Calibration, acceptance, and policy weights

Fit shared temperature $T_c > 0$ on a separate temperature-calibration split by ensemble-mean binary NLL (Guo et al., 2017):

$$p_m^{\mathrm{cal}} = \sigma(z_m/T_{\mathrm{c}}), \qquad v_\phi = \frac{1}{M}\sum_m p_m^{\mathrm{cal}}, \qquad d_\phi = \sqrt{\frac{1}{M}\sum_m (p_m^{\mathrm{cal}} - v_\phi)^2}. \tag{14}$$

This updates post-processing, not adapters. An independent threshold split reproduces the policy/version, N, decoding, acquisition, edits, updated ranking, and extra-check pipeline. After paid executions, label the highest-scoring still-unexecuted candidate solely for calibration; that terminal label cannot trigger replacement. No remaining candidate means no model-only acceptance. Keep extra checking independent of the threshold. This targets selected rather than random-candidate risk (Geifman & El-Yaniv, 2017); a changed pipeline or deployment distribution requires fresh calibration or justified transfer. Without adequate calibration, use confirmed successes only.

Before threshold labels, fix finite grid Γ, risk target $r_{\max}$, confidence $1-\delta$, minimum count $m_{\min}$, and the other selection parameters. For each threshold, m_γ accepted tasks and k_γ observed failures give the grid-corrected one-sided Clopper–Pearson bound (Clopper & Pearson, 1934):

$$U_\gamma = \begin{cases} 1, & m_\gamma = 0 \text{ or } k_\gamma = m_\gamma, \\ \mathrm{Beta}^{-1}_{1-\delta/|\Gamma|}(k_\gamma + 1, m_\gamma - k_\gamma), & \text{otherwise}, \end{cases} \tag{15}$$

Choose greatest empirical coverage among thresholds satisfying $m_\gamma \geq m_{\min}$ and $U_\gamma \leq r_{\max}$, with a fixed tie rule; if none qualifies, disable model-only acceptance. The exact interpretation requires independent identically distributed task-level trials and a fixed within-task pipeline. Conditional on m_γ, errors are binomial; each bound fails with probability at most $\delta/|\Gamma|$, so the union bound permits data-dependent grid selection. Correlation between thresholds is harmless, but linked task variants are not independent trials. Adaptive cross-task budgets, reused labels across rounds, and shift are not covered. Use fresh task batches and a preset cross-round/domain error allocation for the certificate; otherwise report empirical task-clustered risk.

Query provenance separates confirmed successes, failures, and unexecuted candidates. Known failures never become positive targets, regardless of confidence. At most the highest-scoring unexecuted candidate per task is admitted when its threshold qualifies. In stochastic tasks, confirmation means the stated observed/replicated criterion, not certainty of future success. Raw weights are 1 for confirmed positives/replay and $\omega \in (0, 1]$ for model-only positives. Uniform task sampling followed by uniform within-task sampling uses

$$w_i = \frac{n_x \widetilde{w}_i}{\sum_{j:x_j=x} \widetilde{w}_j}, \qquad \frac{1}{n_x}\sum_{i:x_i=x} w_i = 1. \tag{16}$$

The relative weight of an unexecuted example to a confirmed example of the same task is ω. If a task has only one unexecuted positive, its normalized weight is one. Thus the specified discount is within-task, not an additional global downweighting of pseudo-only tasks. Task-level normalization prevents tasks with many accepted candidates from dominating solely through count. Sampling only represented tasks is an empirical training rule; it does not remove the common-task-distribution assumptions in Appendix G.

Let $\mathcal{H}_t$ be the fixed distribution of policy-token prefixes in this task-balanced training set, with a specified within-trajectory prefix sampling rule. A concrete reference regularizer is

$$\mathcal{R}_{\mathrm{KL}}(\theta, \theta_t) = \mathbb{E}_{(x,h)\sim\mathcal{H}_t} \mathrm{KL}(\pi_{\theta_t}(\cdot \mid x, h) \| \pi_\theta(\cdot \mid x, h)) \,. \tag{17}$$

Keep the sampling-policy logits detached. Compute the categorical KL over the vocabulary or report a fixed approximation consistently across methods. This penalty and replay trade off fitting new targets against retaining previous behavior. Fix update steps or development-based stopping before test evaluation. If $\mathcal{D}_t$ is empty, set $\theta_{t+1} = \theta_t$; otherwise, absent tasks receive no direct positive loss but can change through shared parameters. Additional execution outcomes are excluded from the already-completed verifier update and become eligible only in the next round. Calibration outcomes and model-only labels never enter training replay.

G CONDITIONAL ANALYSIS

G.1 A SINGLE-ROUND IMPROVEMENT BOUND

Fix a task distribution $\mathcal{T}$, an execution kernel $\mathcal{E}$, and a utility $u \in [0,1]$. Define

$$r(x,\tau) = \mathbb{E}_{o\sim\mathcal{E}(\cdot|x,\tau)}u(o), \qquad J(\pi) = \mathbb{E}_{x\sim\mathcal{T},\tau\sim\pi(\cdot|x)}r(x,\tau). \tag{18}$$

For binary tasks, $D_t = J(\pi_{\theta_t})$. Write $\pi_t = \pi_{\theta_t}$ and $v_t = v_{\phi_{t+1}}$ in this section. Condition on the realized learning history, so that the policies, verifier, and selection target are fixed. Randomness in a new task or its execution is still averaged in (18).

The target $\rho_t(\tau \mid x)$ is the conditional trajectory distribution induced by accepted examples and their weights, including successful edits and replay. For a task without positive-weight examples, define $\rho_t(\cdot \mid x) = \pi_t(\cdot \mid x)$ for the analysis. This convention preserves the common task marginal; it does not assert that updating shared policy parameters leaves that task's behavior unchanged. Such interference contributes to the policy error below. Define the predicted target advantage by

$$\widehat{A}_t = \mathbb{E}_{x\sim\mathcal{T}}\left[\mathbb{E}_{\tau\sim\rho_t(\cdot|x)}v_t(x,\tau) - \mathbb{E}_{\tau\sim\pi_t(\cdot|x)}v_t(x,\tau)\right]. \tag{19}$$

Proposition (single-round sufficient condition). Suppose that, for nonnegative $\varepsilon_t^{\mathrm{V}}$ and ε_t^{π},

$$\begin{aligned} \mathbb{E}_{\mathcal{T},\rho_t}|r - v_t| \le \varepsilon_t^{\mathrm{V}}, \quad \mathbb{E}_{\mathcal{T},\pi_t}|r - v_t| \le \varepsilon_t^{\mathrm{V}}, \\ \mathbb{E}_{x\sim\mathcal{T}}\,\mathrm{TV}(\pi_{t+1}(\cdot \mid x), \rho_t(\cdot \mid x)) \le \varepsilon_t^{\pi}, \end{aligned} \tag{20}$$

where $\mathrm{TV}(p,q) = \frac{1}{2}\int |p-q|$. Then

$$J(\pi_{t+1}) - J(\pi_t) \ge \widehat{A}_t - 2\varepsilon_t^{\mathrm{V}} - \varepsilon_t^{\pi}. \tag{21}$$

In particular, a strictly positive right-hand side is sufficient for improvement.

Proof. Add and subtract $J(\rho_t)$ and the two expected verifier scores:

$$\begin{aligned} J(\pi_{t+1}) - J(\pi_t) = J(\pi_{t+1}) - J(\rho_t) + \widehat{A}_t \\ + \mathbb{E}_{\mathcal{T},\rho_t}(r - v_t) - \mathbb{E}_{\mathcal{T},\pi_t}(r - v_t). \end{aligned} \tag{22}$$

Because $0 \le r \le 1$, the difference of its expectations under two conditional distributions is bounded in absolute value by their total variation. One way to see this is to write $r = \int_0^1 \mathbf{1}\{r > s\}\, ds$ and apply the definition of total variation to each level set. Consequently,

$$J(\pi_{t+1}) - J(\rho_t) \ge -\mathbb{E}_{\mathcal{T}}\,\mathrm{TV}(\pi_{t+1}, \rho_t) \ge -\varepsilon_t^{\pi}.$$

Each remaining signed error term in (22) is at least $-\varepsilon_t^{\mathrm{V}}$ by (20). Summing proves (21), and substituting binary utility gives (7). □

The two verifier conditions cover both ordinary policy outputs and the selected distribution. Good average prediction on an unrelated pool need not control errors in the selected tail. Likewise, a low training loss does not establish population-level policy approximation. These are conditions on the realized update, not information available to the training algorithm. They allow finite support, imperfect optimization, replay shifts, and cross-task interference to appear in the errors rather than assuming them away. They concern one fixed utility and environment distribution: passing a training checker does not transfer the bound to a stronger hidden checker without new error estimates for that checker.

G.2 SELECTION TARGETS AND POLICY DISTILLATION

Let $\mu_t(\tau \mid x)$ be the proposal distribution of available trajectories, and let $\widetilde{w}_t(x,\tau) \ge 0$ combine acceptance and source weights. Where $Z_t(x) > 0$, define

$$\rho_t(\tau \mid x) = \frac{\mu_t(\tau \mid x)\widetilde{w}_t(x,\tau)}{Z_t(x)}, \qquad Z_t(x) = \mathbb{E}_{\tau\sim\mu_t(\cdot|x)}\widetilde{w}_t(x,\tau). \tag{23}$$

With only original rollouts, the population proposal is π_t; with edits and replay it is their corresponding mixture. A realized finite pool instead gives an empirical proposal. The theorem applies

to a specified target, including an empirical target on a finite task population; a held-out-population interpretation requires assumptions on that population, not merely the empirical fit.

For a fixed ρ_t, minimizing its expected negative log policy likelihood minimizes $\mathbb{E}_{\mathcal{T}}\mathrm{KL}(\rho_t\|\pi_\theta)$ up to the target entropy. This is the policy-distillation perspective also used in expert iteration (Anthony et al., 2017). If this *population forward KL* is at most κ_t, Pinsker's inequality and Jensen's inequality imply

$$\mathbb{E}_{\mathcal{T}}\,\mathrm{TV}(\pi_\theta,\rho_t) \le \mathbb{E}_{\mathcal{T}}\sqrt{\tfrac{1}{2}\mathrm{KL}(\rho_t\|\pi_\theta)} \le \sqrt{\kappa_t/2}.$$

The token-level reference penalty in (6) is a different quantity. It stabilizes an update on sampled prefixes; it does not certify the forward-KL or total-variation condition on complete trajectories. The sharpening analysis of Huang et al. (2025) provides a complementary perspective on concentrating generation on high-quality sequences, while the bound here explicitly retains verification and target-fitting errors.

G.3 Outcome prediction, sampling, and diagnostic scope

On a fixed input distribution $P(x,\tau)$, let $p_*(o \mid x,\tau)$ be the true outcome law. Whenever the expectations are finite,

$$\mathbb{E}_{P,p_*}[-\log q(o \mid x,\tau)] = \mathbb{E}_P[H(p_*(\cdot \mid x,\tau)) + \mathrm{KL}(p_*\|q)]\,. \tag{24}$$

For fixed P and environment, lower population log loss means better outcome-distribution fit. Lower predictive entropy alone can instead reflect overconfidence. Accordingly, audits pair log loss with Brier score and ranking. Fieldwise losses have this interpretation on each field's observed-data distribution; selective observation does not establish calibration on unobserved fields.

For an ideal Bayes predictor with fixed target Y, prior information H_t, and feedback E, the optimal log-loss reduction is

$$H(Y \mid H_t) - H(Y \mid H_t, E) = I(Y; E \mid H_t). \tag{25}$$

All terms use the same joint law. Learned cross-entropy additionally reflects approximation error; lower entropy may be overconfidence. Matched-budget comparisons test the utility of action-dependent feedback, not a universal information law.

With independent per-task success probability p_x,

$$C_x(N) = 1-(1-p_x)^N, \qquad C(N) = \mathbb{E}_{x\sim\mathcal{T}}[1-(1-p_x)^N]. \tag{26}$$

Generally $C(N) \neq 1-(1-D)^N$; dependent candidates require their joint distribution (Huang et al., 2025). Conditional on best correct score z and m independent incorrect scores with continuous CDF F_-, none exceeds z with probability $F_-(z)^m$. Both z and the correct-candidate count vary with N, so recognition need not decrease. At a fixed policy,

$$\Delta_N G = R\,\Delta_N C + C\,\Delta_N R + \Delta_N C\,\Delta_N R. \tag{27}$$

With zero successes in m independent attempts on one fixed task, the one-sided upper confidence bound at error level α is $1-\alpha^{1/m}$; post-selection of failed tasks needs separate accounting. Finite non-observation cannot prove $p_x = 0$.

Clean-update U (Appendix A.4), audited G, and predicted advantage $\widehat{A}$ are distinct. Comparing true/score maximizers gives argmax regret at most 2ϵ under uniform pool error $|v-r| \le \epsilon$, not a policy-learning guarantee. Population error/TV conditions can be checked directly in enumerable environments with known distributions; token loss, sampled-prefix KL, and observed pool maxima do not certify them. Appendix B.2 tests limited operating conditions, not an improvement identity or indefinite growth.