When the Verifier Is Silent: Process Advantages for Agentic Coding

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning for agentic software engineering benefits from strict executable verifiers, but their binary feedback is sparse. In group-relative policy optimization, a task provides no outcome-based learning signal when all sampled rollouts receive the same verifier outcome, despite substantial differences in their process quality. Adding dense process rewards creates a structural tension: small coefficients may vanish under outcome normalization, whereas large coefficients can overturn the verifier-induced success or failure ordering. We introduce KAT, a verifier-preserving process-supervision method that makes process signals subordinate by construction. KAT partitions rollouts by executable outcome, standardizes process signals within each outcome stratum, and combines bounded process advantages with margin-scaled outcome advantages. This guarantees that no judge output or nonnegative process weight can rank an unverified trajectory above a verified one, while outcome-homogeneous groups still provide learning signals whenever process quality varies. KAT derives process signals from a frozen software-engineering lifecycle rubric covering reproduction, root-cause diagnosis, post-fix and regression validation, and tool use. An LLM judge performs evidence-linked structured matching, while deterministic components validate evidence and aggregate scores using reliability-based weights. Starting from Qwen3.6-35B-A3B, KAT improves pass@1 from 64.40% to 69.40% on SWE-bench Verified, 57.00% to 63.00% on SWE-bench Multilingual, 40.63% to 45.96% on SWE-bench Pro, and 32.02% to 41.02% on Terminal-Bench 2.1. It also transfers to SciCode, improving pass@1 from 37.53% to 44.20%. Controlled experiments show that KAT provides usable supervision and outperforms outcome-only and additive-reward alternatives. Models will be publicly available.

1 Introduction

Large language models are increasingly deployed as autonomous coding agents that inspect repositories, edit source files, invoke development tools, and validate their own changes (Zeng et al., 2026; Xu et al., 2026; Yang et al., 2025a). Reinforcement learning is particularly attractive in this setting because software engineering provides executable feedback (Le et al., 2022; Gehring et al., 2025): a repository-level verifier can determine whether a generated patch satisfies the required behavior. However, an agentic coding episode may contain hundreds of interdependent decisions, including reproducing a reported failure, localizing its cause, implementing a fix, debugging the implementation, and running targeted and regression tests. A single terminal reward must therefore assign credit across a long, heterogeneous, and largely unlabeled trajectory.

The most trustworthy supervision is a strict executable outcome. A patch succeeds only if it resolves the reported failure (fail-to-pass) and preserves previously correct behavior (pass-to-pass) (Jimenez et al., 2024). This signal is precise but also sparse. The sparsity is especially problematic for group-relative policy optimizers (Shao et al., 2024; Zheng et al., 2025), which construct advantages from reward differences among multiple rollouts sampled for the same task. When all rollouts in a group fail, the binary outcome reward is constant, so the corresponding group-relative advantage vanishes identically (Yu et al., 2025). The update is then empty even if one rollout reproduced the failure, localized the relevant component, and passed a subset of the target tests, whereas another edited unrelated code and never attempted validation. A substantial fraction of groups in our distribution are outcome-homogeneous, so outcome-only optimization discards considerable rollout compute.

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.

A common response is to define an additive reward composed of outcome and process components:

$$R_i = R_i^{\text{out}} + \eta R_i^{\text{proc}}, \tag{1}$$

where η controls the influence of process supervision. However, this formulation assigns two conflicting roles to a single coefficient. To preserve the verifier-induced ordering, η must be small enough to prevent a high-scoring failed rollout from outranking a successful one; yet in mixed-outcome groups, outcome variance can suppress process differences after normalization, requiring a larger η to retain process sensitivity. Additive mixing therefore couples verifier fidelity and process sensitivity through a scale-dependent hyperparameter. Our starting principle is that process supervision should refine only comparisons left unresolved by the verifier. We thus introduce *outcome-stratified process advantages*: rollouts are partitioned into success and failure strata, process signals are standardized within each stratum, and their weighted combination is bounded before being added to a margin-scaled outcome advantage. This separates cross-outcome ranking from within-outcome refinement, guaranteeing that no judge output or nonnegative process weight can cause an unverified trajectory to outrank a verified one. Meanwhile, an outcome-homogeneous group still provides a learning signal whenever its combined process quality varies. Accordingly, KAT does not treat the process judge as a value function or require its scalar output to represent absolute trajectory quality; it uses the signal only to rank trajectories with the same observed verifier outcome, reducing the consequences of judge miscalibration.

We construct the process signal using three components. First, a scenario-bound lifecycle rubric evaluates pre-fix reproduction, root-cause diagnosis, targeted post-fix validation, regression validation, and policy-attributable tool use. Its schema is shared across tasks, while its criteria are grounded in task-specific failure scenarios extracted from each problem statement. Second, judgment is separated from scoring: an LLM judge emits evidence-linked rule matches, while deterministic checks validate the cited evidence and compile accepted matches into ordinal lifecycle states and fixed utilities. The judge therefore does not directly determine the reward scale. Third, conformance calibration weights each rubric dimension by its measured reliability on a held-out audit suite containing positive, negative, contrastive, temporal, and invariance cases; dimensions below a reliability threshold are excluded from training.

Starting from Qwen3.6-35B-A3B (Qwen Team, 2026b), we train KAT and evaluate it on six SWE and agentic benchmarks. Relative to the base model, KAT improves pass@1 by 5.00 percentage points on SWE-bench Verified, 6.00 on SWE-bench Multilingual, 5.33 on SWE-bench Pro, 9.00 on Terminal-Bench 2.1, 1.22 on PinchBench, and 6.67 on SciCode. The gains on Terminal-Bench 2.1 and SciCode suggest that the learned reproduce-diagnose-validate discipline transfers beyond SWE tasks. Controlled ablations further show that outcome stratification, bounded process combination, scenario binding, and reliability calibration each contribute to the final performance.

Our contributions are as follows:

- We identify a structural limitation of additive process rewards in group-relative optimization: one scale-dependent coefficient must both preserve the strict verifier ordering and retain process contrasts after group normalization.
- We introduce an outcome-stratified and bounded process advantage that preserves the verifier ordering for arbitrary judge outputs and nonnegative process weights (Proposition 1), while recovering nonzero supervision from outcome-homogeneous groups whenever their combined process quality varies (Proposition 2).
- We develop an auditable process-evaluation pipeline based on scenario-bound lifecycle criteria, evidence-linked judgment, deterministic state compilation, and reliability-aware weighting.
- We evaluate KAT on 6 SWE and general agent benchmarks and use controlled ablations, credit-coverage measurements, continuation experiments to analyze the source of the gains.

2 Related Work

Process and rubric supervision. To supervise how solutions are reached, reward models learn from human judgments of intermediate reasoning steps (Uesato et al., 2022; Lightman et al., 2024). To automate such feedback, methods derive process labels from sampled continuations (Wang et al., 2024; Luo et al., 2024) or estimate intermediate values and advantages using Monte Carlo

or temporal-difference targets (Kazemnejad et al., 2025; Choudhury, 2025; Xi et al., 2026). Implicit reward methods instead recover token- or action-level feedback from trajectory outcomes or preferences without explicit step annotations (Yuan et al., 2025; Cui et al., 2026; Liu et al., 2026b). Because such targets are hard to identify over long interactive horizons, a complementary line makes the criteria explicit instead of inferring them: learned checklist evaluators guide web action selection (Chae et al., 2025) and action- and task-level evaluators guide iterative GUI data collection (Xiao et al., 2025), though both apply the assessment at inference or data-collection time rather than as a training-time credit signal. Rubric-based on-policy rewards (Gunjal et al., 2026) and repository-aware patch evaluation (Raghavendra et al., 2026) bring explicit criteria into optimization. KAT adopts this criterion-based view but differs in how criteria are grounded and weighted: each criterion is bound to failure scenarios extracted once per task, so validation-like behavior unrelated to the reported issue earns no credit, and dimension weights are set by audited reliability rather than by hand, with criteria below a reliability floor excluded from the training signal altogether. A frozen judge supplies only evidence-linked semantic matches, while applicability, temporal and citation validity, state compilation, and arithmetic remain deterministic (Section 3.2).

Advantage construction. For policy updates on verifiable tasks, GRPO and GSPO compare rollouts within task groups (Shao et al., 2024; Zheng et al., 2025), and Stratified GRPO (Zhu et al., 2025) conditions normalization on structural strata. Such estimators are uninformative when a group is outcome-homogeneous, and DAPO (Yu et al., 2025) responds by filtering these groups out; KAT moves in the opposite direction, treating them as the dominant share of sampled experience and recovering supervision from them. When process feedback is added, its interaction with outcomes also matters: ReNorm (Xu et al., 2025) shifts step rewards by outcomes before advantage, PAPO normalizes process scores only for correct responses (Tan et al., 2026), PASC-GRPO applies outcome-dependent one-sided clipping (Liu et al., 2026a), and SWE-TRACE enforces success–failure separation through a trajectory-reward margin (Han et al., 2026). KAT differs on both sides of this interaction. It standardizes each process signal within its observed outcome stratum—both strata, not only the successful one—so that failed rollouts are ranked against other failures and homogeneous groups still produce gradient signal. And rather than penalizing violations of the outcome ordering, it bounds the combined process advantage before adding it to the margin-scaled outcome advantage, which makes the ordering hold by construction for every judge realization and every calibrated weight configuration (Section 3.3).

3 METHOD

3.1 PROBLEM SETUP AND DESIGN PRINCIPLES

A task x consists of a problem statement, a repository at a fixed commit, and a sandbox execution environment. A policy π_θ interacts with the repository through search, inspection, editing, and test-execution tools. It produces a variable-length trajectory $\tau = (o_0, a_0, \ldots, a_{T-1}, o_T)$ and a final patch $\Delta(\tau)$. At each update, we independently sample a group $\mathcal{G}_x = \{\tau_i\}_{i=1}^{G}$ from the behavior policy $\pi_{\theta_{\text{old}}}$, where G is the group size. Let p_i^{F2P} and p_i^{P2P} denote the fractions of applicable fail-to-pass and pass-to-pass tests passed by rollout i. Its strict verifier outcome is:

$$y_i = \mathbf{1}\big[p_i^{\text{F2P}} = 1 \wedge p_i^{\text{P2P}} = 1\big] \in \{0, 1\}, \tag{2}$$

where unavailable test categories are omitted from the conjunction.

Deterministic trustworthiness checks remove rollouts whose outcomes are unobservable because of infrastructure failures. We denote the retained group by $\mathcal{G}_x^{\text{valid}}$ and its size by G'_x. Unless stated otherwise, all statistics below are computed only over the retained group.

KAT restores this information under three design principles:

1. Process criteria are defined by a frozen lifecycle rubric and bound to task-specific failure cases.
2. The LLM judge performs evidence-linked semantic matching, whereas applicability, structural validation, state compilation, and arithmetic are deterministic.
3. Process quality is compared only among trajectories with the same strict verifier outcome, and its final contribution is bounded.

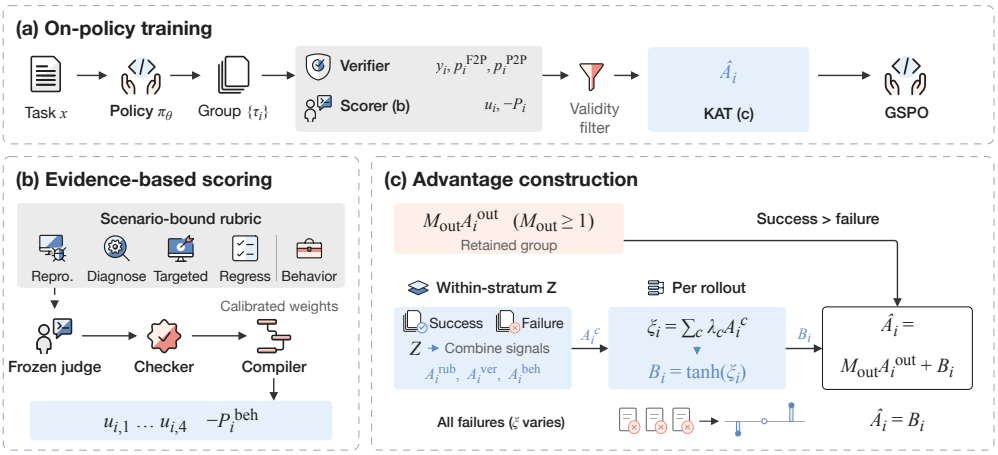


Figure 1: Overall design concept of advantage construction.

We rely on two operational conditions, described in Section 3.4: retained verifier outcomes must reflect policy behavior rather than infrastructure faults, and rubric criteria must be decidable without access to the reference patch. The substantive empirical assumption is that, conditional on the task and current verifier outcome, higher lifecycle quality predicts better future progress.

3.2 Scenario-Bound Lifecycle Rubric

Frozen ontology and task-specific binding. We define four ordinal lifecycle dimensions $\mathcal{D} = \{\mathrm{rep}, \mathrm{diag}, \mathrm{tval}, \mathrm{reg}\}$, corresponding to pre-fix reproduction, root-cause diagnosis, post-fix validation of the reported failures, and regression validation beyond those failures. Each dimension $d \in \mathcal{D}$ owns a disjoint criterion group $\mathcal{K}_d$, and a fifth disjoint group $\mathcal{K}_{\mathrm{beh}}$ captures non-ordinal execution pathologies. We write $\mathcal{K} = \big(\cup_{d\in\mathcal{D}} \mathcal{K}_d\big) \cup \mathcal{K}_{\mathrm{beh}}$ and index individual criteria by $k \in \mathcal{K}$. The ontology, state-transition rules, and utility maps are fixed before reinforcement learning begins.

Because a task-agnostic rubric may reward validation unrelated to the issue, a fixed binder $\mathcal{B}_\psi$ extracts failure scenarios $\mathcal{S}_x = \{s_m^x\}_{m=1}^{M_x}$ once per task. Each scenario $s_m^x = (t_m, \eta_m^{\mathrm{obs}}, \eta_m^{\mathrm{exp}}, c_m)$ specifies the trigger, reported and expected behaviors, and explicit constraints. Each field must cite the problem statement; ungrounded or inapplicable criteria are masked. We fix the binder's prompt, schema, decoding configuration, and model version, caching its output across all task rollouts to prevent rollout-dependent criteria.

Evidence-linked judgment. The frozen judge J_ϕ receives the task, scenarios $\mathcal{S}_x$, rubric, chronological trajectory τ_i, patch $\Delta(\tau_i)$, harness results $(p_i^{\mathrm{F2P}}, p_i^{\mathrm{P2P}})$, and execution metadata, without the reference patch or hidden-test bodies. For each criterion $k \in \mathcal{K}$, it emits a binary semantic verdict $b_{i,k}$ with supporting turn identifiers $e_{i,k}$. Behavioral criteria use reversed polarity: $b_{i,k} = 1$ denotes an *observed* pathology.

A deterministic checker then verifies that the cited turns exist and satisfy criterion-specific temporal and structural requirements. Reproduction evidence must precede the first edit to a non-test source file; post-fix validation must follow that edit; regression evidence must extend beyond the bound failure scenarios; and claimed environment blockers must be corroborated by execution metadata.

Let $m_{x,k} \in \{0,1\}$ denote binder-assigned applicability and $v_{i,k} \in \{0,1\}$ the evidence-check result. The accepted verdict is $\widetilde{b}_{i,k} = m_{x,k} b_{i,k} v_{i,k}$, and dimension-level applicability is $m_{x,d} = \max_{k\in\mathcal{K}_d} m_{x,k} \in \{0,1\}$ for $d \in \mathcal{D}$. An unaccepted validation criterion is classified as `not_attempted`, `agent_error`, or `environment_blocked`. The first two remain policy-attributable; corroborated environment blockers are exempt from omission penalties, so unreliable environments do not discourage validation attempts.

Deterministic state compilation. For each dimension $d \in \mathcal{D}$, the deterministic compiler maps accepted criteria $\{\widetilde{b}_{i,k}\}_{k\in\mathcal{K}_d}$ to the highest satisfied ordinal state z_i^d; nested state definitions make this

choice unique. Cause-aware blocked states precede `not_attempted`. We compile reproduction and diagnosis separately, allowing diagnosis credit when dynamic reproduction is unavailable or unsuccessful. Each state has fixed monotone utility $u_{i,d} := u_d(z_i^d) \in [0,1]$. Inapplicable dimensions are masked with $m_{x,d} = 0$, not assigned zero utility.

Conformance calibration. We audit the frozen binder-judge-compiler pipeline on human-adjudicated positive, negative, contrastive, temporal, and invariance cases. If N_d^{corr} of N_d^{tot} audited trajectories receive the correct compiled state for dimension d, a Beta prior (α_0, β_0) gives the smoothed reliability $q_d = (N_d^{\mathrm{corr}} + \alpha_0)/(N_d^{\mathrm{tot}} + \alpha_0 + \beta_0)$. We calibrate the dimension weight as:

$$\widetilde{w}_d = w_d\, f(q_d), \qquad f(q) = \max\left(0, \frac{q - q_0}{1 - q_0}\right), \tag{3}$$

where $w_d > 0$ is a prior importance weight and $q_0 \in (0,1)$ is the reliability floor. We apply the same calibration to each behavioral criterion, giving $\widetilde{w}_k = w_k f(q_k)$. Criteria below the floor contribute only to diagnostic reporting. The effective dimension mask is $\widetilde{m}_{x,d} = m_{x,d} \cdot \mathbf{1}[\widetilde{w}_d > 0]$, which we use throughout scoring and advantage construction. For reporting and scalar-reward baselines, we define the lifecycle score and capped behavioral penalty as

$$S_i^{\mathrm{life}} = \frac{\sum_{d \in \mathcal{D}} \widetilde{m}_{x,d} \cdot \widetilde{w}_d \cdot u_{i,d}}{\sum_{d \in \mathcal{D}} \widetilde{m}_{x,d} \cdot \widetilde{w}_d + \varepsilon} \in [0,1], \quad P_i^{\mathrm{beh}} = \min\left(P_{\max}, \sum_{k \in \mathcal{K}_{\mathrm{beh}}} \widetilde{w}_k\, \widetilde{b}_{i,k}\right) \in [0, P_{\max}]. \tag{4}$$

The cap $P_{\max}$ prevents P_i^{beh} from becoming an implicit trajectory-length penalty. Scalar-reward baselines use $R_i^{\mathrm{rub}} = \kappa S_i^{\mathrm{life}} - P_i^{\mathrm{beh}}$ with a fixed scale $\kappa > 0$, whereas KAT retains the lifecycle dimensions as separate signals.

3.3 Verifier-Preserving Group Advantage

Within-stratum standardization. We partition retained rollouts into $\mathcal{G}_x^{(g)} = \{\tau_i \in \mathcal{G}_x^{\mathrm{valid}} : y_i = g\}$ for $g \in \{0,1\}$ and compare process signals only within each outcome stratum. Let $r = (r_i)_{\tau_i \in \mathcal{G}_x^{\mathrm{valid}}}$ be any group-level scalar signal, and let $\mu_g(r)$ and $\sigma_g(r)$ denote its mean and sample standard deviation over $\mathcal{G}_x^{(g)}$. The standardization operator is:

$$\mathsf{Z}_i(r) = \begin{cases} \dfrac{r_i - \mu_{y_i}(r)}{\sigma_{y_i}(r) + \varepsilon}, & \text{if } |\mathcal{G}_x^{(y_i)}| \geq 2 \text{ and } \sigma_{y_i}(r) > \delta_{\mathrm{var}}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The variance gate prevents negligible disagreements from being amplified; singleton, constant, and sufficiently low-variance strata contribute zero.

For lifecycle dimension d, we apply this operator to the utility signal $u_{\cdot,d}$ and define:

$$a_{i,d} = \widetilde{m}_{x,d} \cdot \mathsf{Z}_i(u_{\cdot,d}), \qquad A_i^{\mathrm{rub}} = \frac{\sum_{d \in \mathcal{D}} \widetilde{m}_{x,d}\, \widetilde{w}_d\, a_{i,d}}{\sum_{d \in \mathcal{D}} \widetilde{m}_{x,d}\, \widetilde{w}_d + \varepsilon}. \tag{6}$$

Standardizing dimensions separately prevents an informative contrast in one lifecycle stage from being diluted by dimensions that are constant within the same outcome stratum.

Machine-checkable and behavioral signals. We additionally incorporate machine-checkable test progress and behavioral quality. Fail-to-pass and pass-to-pass rates are standardized separately within each outcome stratum and combined with fixed nonnegative weights α_{F2P} and α_{P2P}:

$$A_i^{\mathrm{ver}} = \frac{\alpha_{\mathrm{F2P}}\, \mathsf{Z}_i(p^{\mathrm{F2P}}) + \alpha_{\mathrm{P2P}}\, \mathsf{Z}_i(p^{\mathrm{P2P}})}{\alpha_{\mathrm{F2P}} + \alpha_{\mathrm{P2P}} + \varepsilon}. \tag{7}$$

Unavailable test categories receive zero weight. Separate standardization prevents larger pass-to-pass suites from dominating target-test progress. Successful rollouts pass all applicable tests, so the variance gate removes their constant test signals; A_i^{ver} operates only in the failure stratum. Behavioral quality $A_i^{\mathrm{beh}} = \mathsf{Z}_i(-P^{\mathrm{beh}})$ rewards fewer policy-attributable pathologies. We combine the process components and bound their total contribution:

$$\xi_i = \lambda_{\mathrm{ver}} A_i^{\mathrm{ver}} + \lambda_{\mathrm{rub}} A_i^{\mathrm{rub}} + \lambda_{\mathrm{beh}} A_i^{\mathrm{beh}}, \qquad B_i = \tanh(\xi_i), \tag{8}$$

where the process coefficients $\lambda_{\text{ver}}, \lambda_{\text{rub}}, \lambda_{\text{beh}} \geq 0$ are fixed. Thus $B_i \in (-1, 1)$ for every realization of the process scores and judge outputs. The post-standardization bound, rather than the raw rubric scale, determines the maximum influence of process supervision.

Outcome advantage and final construction. Let $\bar{y}_x = (1/G'_x) \sum_{\tau_i \in \mathcal{G}_x^{\text{valid}}} y_i$ be the success rate among retained rollouts. We define:

$$A_i^{\text{out}} = \begin{cases} \dfrac{y_i - \bar{y}_x}{\sqrt{\bar{y}_x(1-\bar{y}_x)}}, & 0 < \bar{y}_x < 1, \\ 0, & \bar{y}_x \in \{0, 1\}. \end{cases} \tag{9}$$

The final KAT advantage is:

$$\widehat{A}_i = M_{\text{out}}\, A_i^{\text{out}} + B_i, \quad M_{\text{out}} \geq 1, \tag{10}$$

where M_{out} controls the outcome margin separately from process coefficients $\lambda_\bullet$. The outcome term separates successes from failures; B_i refines within-stratum rankings and supplies the sole advantage in homogeneous groups.

Proposition 1 (Verifier preservation)**.** *Let $\tau_i, \tau_j \in \mathcal{G}_x^{\text{valid}}$ satisfy $y_i = 1$ and $y_j = 0$. Then, for every judge realization, every calibrated rubric-weight configuration, and every nonnegative choice of process coefficients, $\widehat{A}_i - \widehat{A}_j > \max\{0,\ 2(M_{\text{out}} - 1)\} \geq 0$, and in particular $\widehat{A}_i > \widehat{A}_j$.*

Increasing M_{out} widens the cross-stratum margin without changing within- ordering. It preserves observed verifier ordering; within-stratum ranking quality depends on process-signal predictiveness.

Proposition 2 (Non-degenerate homogeneous-group supervision)**.** *Suppose all retained rollouts for task x share the same verifier outcome. If the combined pre-bounding process score ξ_i is nonconstant over $\mathcal{G}_x^{\text{valid}}$, then the final KAT advantage $\{\widehat{A}_i\}$ is nonconstant, and at least one retained rollout receives a nonzero advantage.*

For a homogeneous group, $A_i^{\text{out}} = 0$ and $\widehat{A}_i = \tanh(\xi_i)$. Strict monotonicity of $\tanh$ preserves nonconstancy, so at least one advantage is nonzero. This requires variation in the *combined* score: other components may cancel a varying component. Groups with only constant or variance-gated signals receive zero advantage and are skipped.

3.4 Optimization and Training Controls

Trustworthiness filtering. The fixed validity filter removes rollouts with unobservable outcomes caused by sandbox initialization failures, repository mismatches, evaluator failures, corrupted harness output, or external interruptions. It retains policy-attributable failures, including malformed commands, incorrect test targets, syntax errors, invalid patches, and agent-induced timeouts. Filtering precedes all group statistics and is independent of rubric scores. We skip tasks with $G'_x < 2$ or zero advantages throughout. Singleton strata receive only the outcome term (Eq. (5)). We report filtering and skip rates by cause.

Sequence-level policy optimization. We optimize the policy using GSPO (Zheng et al., 2025). Let $\mathcal{A}_i$ contain the token positions of policy-generated tokens in τ_i; system tokens, user tokens, and tool observations are masked out. The length-normalized sequence-level importance ratio is:

$$\rho_i(\theta) = \exp\left[\frac{1}{|\mathcal{A}_i|} \sum_{t \in \mathcal{A}_i} \left(\log \pi_\theta(\tau_{i,t} \mid \tau_{i,<t}) - \log \pi_{\theta_{\text{old}}}(\tau_{i,t} \mid \tau_{i,<t})\right)\right]. \tag{11}$$

Let $\mathcal{X}_{\text{upd}}$ denote the current batch with valid, nonzero group advantages. We minimize:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{X}_{\text{upd}}|} \sum_{x \in \mathcal{X}_{\text{upd}}} \frac{1}{G'_x} \sum_{\tau_i \in \mathcal{G}_x^{\text{valid}}} \min\left(\rho_i(\theta)\widehat{A}_i, \text{clip}\left(\rho_i(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}}\right) \widehat{A}_i\right). \tag{12}$$

The trajectory-level advantage $\widehat{A}_i$ is shared by all policy-generated tokens in τ_i. Following GSPO, we use asymmetric clipping and no explicit KL penalty.

Table 1: KAT hyperparameters.

Symbol	Value	Symbol	Value
Advantage construction		*Rubric and calibration*	
M_{out}	1.5	$P_{\max}$ / q_0	1.0 / 0.80
λ_{ver} / λ_{rub} / λ_{beh}	0.3 / 0.5 / 0.2	(α_0, β_0)	$(1, 1)$
α_{F2P} / α_{P2P}	0.7 / 0.3	κ / η (baselines)	1.0 / 0.3
δ_{var} / ε	0.05 / 10^{-6}		
Dimension weights $w_d \to \widetilde{w}_d$		*GSPO optimization (no KL)*	
rep / diag	$1.00 \to 0.69$ / $1.50 \to 0.34$	G / batch (tasks) / epochs	8 / 128 / 5
tval / reg	$1.25 \to 0.74$ / $0.75 \to 0.30$	ϵ_{low} / ϵ_{high}	3×10^{-4} / 4×10^{-4}
		μ / KL / reference model	1 / none / none

For a mixed group, the magnitude of the standardized binary outcome advantage satisfies $|A_i^{\text{out}}| \leq \sqrt{G'_x - 1}$, with equality when exactly one rollout differs in outcome from the rest. Since $|B_i| < 1$, the final advantage obeys:

$$|\widehat{A}_i| < M_{\text{out}}\sqrt{G'_x - 1} + 1 \quad \text{(mixed group)}, \qquad |\widehat{A}_i| < 1 \quad \text{(homogeneous group)}. \tag{13}$$

So KAT preserves the scale order of outcome-only optimization while broadening credit coverage.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on six public benchmarks spanning repository-level software engineering, command-line interaction, general agentic tasks, and scientific programming. *SWE-bench Verified* contains 500 human-validated repository-level tasks (Jimenez et al., 2024; Chowdhury et al., 2024). *SWE-bench Multilingual* contains 300 curated tasks from 42 repositories in nine programming languages (Yang et al., 2025b). For *SWE-bench Pro*, we use the 731-instance public split, which emphasizes long-horizon software engineering (Deng et al., 2025). *Terminal-Bench 2.1* contains 89 containerized command-line tasks (Merrill et al., 2026; Terminal-Bench, 2026). *PinchBench* contains N_{PB} real-world OpenClaw tasks, each graded by a deterministic checker, an LLM judge, or both (PinchBench, 2026). *SciCode* contains 80 scientific-programming problems decomposed into 338 subproblems (15 development and 65 test problems); we report subproblem accuracy on the 65 test problems in the with-background setting (Tian et al., 2024).

Training details. We train on SWE-rebench-V2 (Badertdinov et al., 2026). From N_{pool} candidates, we sample eight rollouts per task with Qwen3.6-35B-A3B and retain tasks with $0 < \sum_i y_i < 8$, using initial outcome diversity as the task-selection criterion. This retains about 3,000 tasks. Initial task selection precedes rollout-level filtering (Section 3.4) and permits later homogeneous groups: success probability $1/8$ gives an all-failure probability $(7/8)^8 \approx 0.34$. We train for five epochs with $G = 8$ and 128 tasks per step, filling 1024 containers in one rollout wave. Training comprises 120 steps and 120,000 trajectories; we refresh $\pi_{\theta_{\text{old}}}$ each step ($\mu = 1$) and resample rollouts each epoch. Caching $\mathcal{S}_x$ at pool construction requires 3,000 binder calls alongside 120,000 judge calls. Table 1 lists the fixed hyperparameters. The floor $q_0 = 0.8$ retains all four lifecycle dimensions and reduces the diagnosis weight from 1.50 to 0.34. With $G = 8$ and $M_{\text{out}} = 1.5$, Eq. (13) gives $|\widehat{A}_i| < 4.97$, which is the only a priori constraint on update magnitude.

Evaluation details. All models are evaluated with pass@1 using a single sample per instance, a fixed seed, and a 256K-token context window (or the model maximum, if smaller), with identical scaffolds, tools, and resource limits within each benchmark. SWE-bench Verified, Multilingual, and Pro use Claude Code 2.1.195 (temperature 1.0, top-p 0.95); PinchBench uses OpenClaw 2026.3.13 and Terminal-Bench 2.1 uses Terminus-2 and Claude Code (temperature 0.7, top-p 1.0); SciCode uses temperature 0.6 and top-p 1.0. For Terminal-Bench 2.1 we report the mean of the two scaffolds together with both components, which Section 4.2 shows to disagree substantially. Ablations use SWE-bench Verified, SWE-bench Pro, and SciCode, covering repository repair, long-horizon software engineering, and scientific programming; all variants share the training configuration above and differ only in the advantage construction. Each is trained once under a fixed seed, so the *Full KAT* rows correspond to the checkpoint of Table 2.

Table 2: Performance on 6 coding and agentic benchmarks. Bold marks the best entry; for Terminal-Bench 2.1 it refers to the mean of the two scaffolds shown underneath. Δ uses unrounded scores.

Model	SWE-bench Verified	SWE-bench Multilingual	SWE-bench Pro	Terminal-Bench 2.1 (Terminus-2 / Claude Code)	PinchBench	SciCode	Avg.
Qwen3.5-27B	68.60	57.67	42.13	34.84 (41.57 / 28.10)	90.71	25.58	53.26
Ornith-1.0-35B	55.80	51.67	34.47	35.98 (35.96 / 36.00)	91.62	30.34	49.98
Gemma4-31B	60.60	49.33	32.97	32.59 (30.34 / 34.83)	85.53	33.19	49.04
Qwen3.5-35B-A3B	58.60	47.67	38.03	26.12 (26.44 / 25.80)	88.75	27.73	47.82
Gemma4-26B-A4B	35.80	27.33	9.58	20.94 (27.27 / 14.60)	82.01	30.84	34.42
Qwen3-Coder-30B	31.80	20.67	19.84	13.50 (10.11 / 16.90)	72.30	18.27	29.40
Qwen3.6-35B-A3B (base)	64.40	57.00	40.63	32.02 (34.83 / 29.20)	92.21	37.53	53.97
KAT (ours)	**69.40**	**63.00**	**45.96**	**41.02** (32.60 / 49.44)	**93.43**	**44.20**	**59.50**
Δ *over base*	*+5.00*	*+6.00*	*+5.33*	*+9.00* (−2.23 / +20.24)	*+1.22*	*+6.67*	*+5.54*

Table 3: Ablation of the credit-assignment design. Benchmark rows are pass@1 (%); coverage and reversal rates are measured on training rollouts.

	Base (no RL)	Full KAT	Outcome only	Additive reward	No stratification	No bounding
SWE-bench Verified	64.40	**69.40**	65.20	67.08	66.72	67.46
SWE-bench Pro	40.63	**45.96**	41.35	43.64	43.09	44.05
Avg.	52.52	**57.68**	53.28	55.36	54.91	55.76
Δ over base	—	**+5.16**	+0.76	+2.84	+2.39	+3.24
Coverage (%)	—	79.2	34.1	76.5	77.1	79.0
Reversal (%)	—	**0.0**	0.0	1.4	1.9	2.6

4.2 Main Results

Table 2 compares KAT with seven recent open-weight models in the 26B-35B range (Qwen Team, 2026a;b; Gemma Team, 2026; DeepReinforce Team, 2026; Qwen Team, 2025). KAT attains the highest score on all six benchmarks, with margins of 0.80, 5.33, 3.83, 5.04, 1.22, and 6.67 points over the strongest competing entry in each column. The Δ row isolates the effect of training: KAT improves over its base checkpoint on every benchmark, by 5.54 points on average, and the gain is not concentrated in a single task family. The gains on SWE-bench Multilingual and Pro indicate transfer to multilingual and long-horizon repository tasks, while those on Terminal-Bench 2.1 and SciCode indicate transfer beyond patch generation to command-line interaction and scientific programming. Margins are smallest where headroom is limited (SWE-bench Verified, where Qwen3.5-27B already reaches 68.60; PinchBench, where several models exceed 90) and largest on the less saturated or more interaction-intensive benchmarks. These results establish the strongest performance *within the evaluated model set* and should not be read as an exhaustive comparison.

4.3 Ablation Studies

Credit-assignment design. Table 3 isolates four choices in Eq. (10): *outcome only* sets all process coefficients to zero, leaving $\widehat{A}_i = M_{\text{out}} A_i^{\text{out}}$; *additive reward* group-standardizes $y_i + \eta R_i^{\text{rub}}$, tuning η on a development split; *no stratification* applies Z_i over the whole retained group instead of within $\mathcal{G}_x^{(g)}$; and *no bounding* replaces $B_i = \tanh(\xi_i)$ with $B_i = \xi_i$. *Coverage* measures the fraction of validity-filtered groups with a nonzero advantage; *reversal* measures the fraction of cross-outcome pairs $(y_i = 1, y_j = 0)$ with $\widehat{A}_i \le \widehat{A}_j$. Outcome-only training improves the two-benchmark average by 0.76 points. In homogeneous groups, $\bar{y}_x \in \{0, 1\}$ makes the outcome advantage vanish (Eq. (9)); only 34.1% of groups contribute, about 44 of 128 sampled tasks per step. These batches incur the full rollout cost while receiving supervision from only their mixed-outcome groups. Additive rewards and pooled standardization restore coverage to about 77% and recover part of the performance gap. Removing the bound retains 79.0% coverage but yields a 2.6% reversal rate: an unbounded process term can dominate the outcome advantage. Full KAT combines 79.2% coverage with zero reversals and the largest gain over the base (5.16 points), showing the value of recovering process supervision.

Lifecycle-rubric construction. Table 4 tests four variants: *no lifecycle rubric* sets $\lambda_{\text{rub}} = 0$, retaining test-progress and behavioral signals; *no scenario binding* applies the frozen ontology without $\mathcal{S}_x$; *no calibration* uses prior weights w_d in place of $\widetilde{w}_d$; and *scalar rubric* standardizes the aggregate S_i^{life} of Eq. (4) instead of individual dimensions. Removing lifecycle supervision costs

Table 4: Ablation of the lifecycle rubric. Benchmark rows are pass@1 (%); Δ Avg. is relative to full KAT and computed from unrounded scores.

	Base (no RL)	Full KAT	No lifecycle rubric	No scenario binding	No calibration	Scalar rubric
SWE-bench Verified	64.40	**69.40**	67.02	67.76	68.38	68.02
SWE-bench Pro	40.63	**45.96**	43.57	44.12	44.79	44.51
Avg.	52.52	**57.68**	55.30	55.94	56.59	56.27
Δ Avg.	−5.16	0.00	−2.38	−1.74	−1.10	−1.42
Coverage (%)	—	79.2	67.8	79.5	80.4	73.6

Table 5: Validity-filtered group shares and nonzero-advantage rates (%).

Group	Share	Nonzero adv. Outcome only	Nonzero adv. KAT
All failure	58.4	0.0	71.3
Mixed outcome	34.1	100.0	100.0
All success	7.5	0.0	46.8
Overall	100.0	34.1	79.2

Table 6: Continuation by within-task lifecycle-quality quartile (%).

Quartile	Resolve rate	Applicable tests passed
Q1 (lowest)	13.2	61.7
Q2	17.9	66.9
Q3	23.8	72.8
Q4 (highest)	31.4	79.5

Table 7: Compiled-state accuracy on the held-out audit set (400 cases per dimension).

	Reproduction	Diagnosis	Targeted validation	Regression validation	Behavioral
Accuracy (%)	94.00	84.75	92.00	88.25	95.25
q_d	0.938	0.846	0.918	0.881	0.950
$f(q_d)$	0.689	0.229	0.590	0.403	0.751

2.38 points, the largest loss among these variants. Test-progress and behavioral signals still distinguish trajectories in many homogeneous groups, retaining 67.8% coverage compared with 34.1% for outcome-only training. Scenario binding adds 1.74 points at similar coverage, consistent with more task-relevant comparisons rather than more contributing groups. Calibration adds 1.10 points despite slightly reducing coverage, showing that additional variation need not improve supervision. Dimension-wise standardization gains 1.42 performance points and 5.6 coverage points over scalar aggregation by preserving contrasts that pre-aggregation can dilute. Together, these results distinguish the amount of supervision from the quality of within-stratum rankings.

4.4 Analysis

Coverage (Table 5). All-failure groups constitute 58.4% of retained groups and are the largest source of experience discarded by outcome-only training. KAT obtains nonzero advantages from 71.3% of them, raising overall coverage from 34.1% to 79.2%. Constant signals or those below δ_{var} still yield zero advantages (Eq. (5)). Contributing groups increase 2.32×, while the median gradient norm changes by 1.11×, consistent with broader use of experience rather than a proportionate increase.

Predictiveness (Table 6). We score failed held-out trajectory prefixes without reference-patch access and sample eight continuations each. Resolve rates rise across within-task quality quartiles, with an 18.2-point Q4–Q1 gap and Spearman correlation 0.31 (task-bootstrap 95% CI $[0.25, 0.37]$). This supports the predictive assumption in Section 3.1 without establishing causality.

Reliability (Table 7). The frozen pipeline is more accurate on observable reproduction, validation, and behavioral evidence than on semantic diagnosis. Calibration reflects this difference in the training weights: $f(q_{\text{diag}}) = 0.23$ versus $f(q_{\text{rep}}) = 0.69$. Deterministic evidence checks reject 8.6% of positive verdicts; the pipeline distinguishes `agent_error` from `environment_blocked`.

5 Conclusion

We introduced KAT, which combines a scenario-bound lifecycle rubric, evidence-checked scoring, and a bounded process term standardized within outcome strata. By construction, process quality refines within-stratum rankings but cannot reverse a verifier outcome, while homogeneous groups still yield gradient signal. Experiments show consistent gains over the base model across six benchmarks and substantially broader credit coverage. The process signal remains predictive rather than causal, judge reliability may shift across distributions, and the frozen rubric limits adaptation to evolving policy behavior. Future work can extend this to areas lacking rigorous verification mechanisms.

AI USE STATEMENT

We used generative AI tools for language polishing and to improve the readability of the manuscript, and as coding assistants during implementation. AI tools were not used to generate research ideas, experimental data, or results. Language models also appear as components of the proposed method itself, namely the frozen scenario binder and rubric judge described in Section 3.2; their prompts, schemas, and model versions are fixed and documented there. The authors reviewed all AI-assisted edits and take full responsibility for the final content.

REFERENCES

Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, and Alexander Golubev. Swe-rebench v2: Language-agnostic swe task collection at scale, 2026. URL https://arxiv.org/abs/2602.23866.

Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-Shepherd: Advancing PRMs for reinforcing web agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 63314–63356. Curran Associates, Inc., 2025. doi: 10.52202/085713-2127. URL https://doi.org/10.52202/085713-2127.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *CoRR*, abs/2502.10325, 2025. doi: 10.48550/arXiv.2502.10325. URL https://doi.org/10.48550/arXiv.2502.10325.

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026, 2026. URL https://openreview.net/forum?id=9SkkifLopZ.

DeepReinforce Team. Ornith-1.0-35B: Agentic coding, open to all, 2026. URL https://deep-reinforce.com/ornith_1_0.html.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can AI agents solve long-horizon software engineering tasks? *CoRR*, abs/2509.16941, 2025.

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

Gemma Team. Gemma 4 technical report. *CoRR*, abs/2607.02770, 2026.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/cfd7bee7a651ee9af525098ef67a9e45-Abstract-Conference.html.

Hao Han, Jin Xie, Xuehao Ma, Weiquan Zhu, Ziyao Zhang, ZhiLiang Long, Hongkai Chen, and Qingwen Ye. SWE-TRACE: Optimizing long-horizon SWE agents through rubric process reward models and heuristic test-time scaling. *CoRR*, abs/2604.14820, 2026. doi: 10.48550/arXiv.2604.14820. URL https://arxiv.org/abs/2604.14820.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *ICLR*. OpenReview.net, 2024.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

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. Curran Associates, Inc., 2022. doi: 10.52202/068431-1549. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*. OpenReview.net, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Runxuan Liu, Xianhao Ou, Xinyan Ma, Jiyuan Wang, Jiafeng Liang, Jiaqi Li, Tao He, Zheng Chu, Rongchuan Mu, Zekun Wang, Baoxin Wang, Dayong Wu, Ming Liu, Shijin Wang, Guoping Hu, and Bing Qin. Graph reasoning paradigm: Structured and symbolic reasoning with topology-aware reinforcement learning for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35859–35881. Association for Computational Linguistics, 2026a. doi: 10.18653/v1/2026.acl-long.1660. URL https://aclanthology.org/2026.acl-long.1660/.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, pp. 129271–129291, 2026b. URL https://openreview.net/forum?id=ooROvpmxMV.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *CoRR*, abs/2406.06592, 2024. doi: 10.48550/arXiv.2406.06592. URL https://arxiv.org/abs/2406.06592.

Mike A. Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, Estefany Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis-Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *CoRR*, abs/2601.11868, 2026.

PinchBench. Pinchbench: Real-world benchmarks for AI coding agents, 2026. URL https://github.com/pinchbench/skill.

Qwen Team. Qwen3-Coder: Agentic coding in the world, 2025. URL https://qwenlm.github.io/blog/qwen3-coder/.

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

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Mohit Raghavendra, Anisha Gunjal, Bing Liu, and Yunzhong He. Agentic rubrics as contextual verifiers for SWE agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15265–15290. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.697. URL https://aclanthology.org/2026.acl-long.697/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Zelin Tan, Zhouliang Yu, Bohan Lin, Zijie Geng, Hejia Geng, Yudong Zhang, Mulei Zhang, Yang Chen, Shuyue Hu, Zhenfei Yin, Chen Zhang, and Lei Bai. PAPO: Stabilizing rubric integration training via decoupled advantage normalization. *CoRR*, abs/2603.26535, 2026. doi: 10.48550/arXiv.2603.26535. URL https://arxiv.org/abs/2603.26535.

Terminal-Bench. Terminal-bench 2.1, 2026. URL https://www.tbench.ai/news/terminal-bench-2-1.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu A. Huerta, and Hao Peng. Scicode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems 37*, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275, 2022. doi: 10.48550/arXiv.2211.14275. URL https://arxiv.org/abs/2211.14275.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195. ACM, 2026. doi: 10.1145/3774904.3792551. URL https://doi.org/10.1145/3774904.3792551.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, Shuai Ren, Yafei Wen, Xiaoxin Chen, Aojun Zhou, and Hongsheng Li. UI-Genie: A self-improving approach for iteratively boosting MLLM-based mobile GUI agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 150376–150411. Curran Associates, Inc., 2025. doi: 10.52202/085713-5028. URL https://doi.org/10.52202/085713-5028.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Peiran Xu, Zhuohao Li, Xiaoying Xing, Guannan Zhang, Debiao Li, and Kunyu Shi. Hybrid reward normalization for process-supervised non-verifiable agentic tasks. *CoRR*, abs/2509.25598, 2025. doi: 10.48550/arXiv.2509.25598. URL https://doi.org/10.48550/arXiv.2509.25598.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025b. doi: 10.52202/085713-3239. URL https://doi.org/10.52202/085713-3239.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38*, 2025. doi: 10.52202/085713-3775. URL https://doi.org/10.52202/085713-3775.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73511–73525. PMLR, 2025. URL https://proceedings.mlr.press/v267/yuan25c.html.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *CoRR*, abs/2507.18071, 2025. doi: 10.48550/arXiv.2507.18071. URL https://arxiv.org/abs/2507.18071.

Mingkang Zhu, Xi Chen, Bei Yu, Hengshuang Zhao, and Jiaya Jia. Stratified GRPO: Handling structural heterogeneity in reinforcement learning of LLM search agents. *CoRR*, abs/2510.06214, 2025. doi: 10.48550/arXiv.2510.06214. URL https://doi.org/10.48550/arXiv.2510.06214.