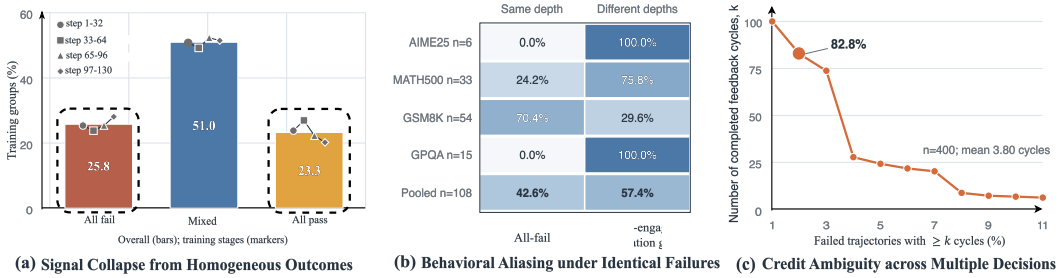
CALIPER: LOCAL INTERVENTION ACROSS REASONING AND TOOL FEEDBACK WITH ADAPTIVE SAMPLING AND SCOPE-ALIGNED UPDATES FOR TOOL-AUGMENTED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic RL commonly samples, compares, and updates full trajectories. Under sparse terminal rewards, more rollouts need not yield more useful supervision: full-trajectory generation consumes the budget, one outcome conflates multiple behavioral changes, and trajectory-level advantages may update decisions unrelated to the observed outcome change. This problem is pronounced in tool-integrated reasoning (TIR), where local changes arise both from autonomous decisions and from responses to newly received tool feedback. The challenge is to choose where to spend computation, which local changes merit comparison, and how to turn their evidence into localized training credit. We propose CALIPER. Under a fixed rollout budget, CALIPER first allocates a portion to full-trajectory exploration, with resulting trajectories serving as training samples and online probes of the current policy. From failed trajectories, CALIPER identifies two types of candidate positions, autonomous model decisions and tool-feedback-driven responses, and estimates their per-compute value from historical success rates, uncertainty, and generation cost. It then adaptively allocates the remaining budget between full exploration and local validation. For selected positions, CALIPER preserves the preceding interaction history and tool state while resampling only subsequent behaviors, yielding strictly verifiable same-state comparisons. During training, CALIPER combines global trajectory advantages with reliability-calibrated local Leave-One-Out (LOO) credit, restricts local credit to the regions where behavior actually changes, and uses negative local evidence to selectively strengthen policy updates. Even when all full trajectories fail, local comparisons can provide non-zero and spatially localized training signals. Across tool-augmented reasoning benchmarks, CALIPER overall outperforms GRPO and other strong baselines, demonstrating improved training-information efficiency under limited rollout budgets.

1 INTRODUCTION



(a) Signal Collapse from Homogeneous Outcomes **(b) Behavioral Aliasing under Identical Failures** **(c) Credit Ambiguity across Multiple Decisions**

Figure 1: Limitations of Terminal-Outcome Supervision in Tool-Augmented RL.

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.

In tool-augmented reasoning, agents decide both during reasoning and after receiving tool feedback (Feng et al., 2026; Chen et al., 2025; Song et al., 2025; Cheng et al., 2026). Outcome-based reinforcement learning, however, assigns terminal rewards to complete interactions; group-based methods such as GRPO (Shao et al., 2024) compare full trajectories for the same task. Homogeneous groups provide no relative signal, while heterogeneous outcomes still cannot identify which intermediate decisions caused the difference. Failed trajectories therefore contain potentially useful reached states, but retrying them competes with generating new full trajectories under a fixed rollout budget (Guo et al., 2026; Ren et al., 2026; Ahmadian et al., 2024). **We ask: under this budget, how can failed trajectories provide comparable decision-level evidence that updates only the behavior being tested?**

Prior work resamples from intermediate states, adapts branch placement, or refines within-trajectory credit, with recent methods combining branch allocation and local credit. Yet a feasible fork is not necessarily worth retrying. Useful retries must justify their compute cost, preserve the same pre-fork tool observations and consumed budgets for valid comparison, and restrict local credit to changed post-fork behavior. **Thus, failed-state retries are useful only when sampling value, comparison basis, and credit scope are aligned.**

Our analysis of Plain GRPO rollout logs and evaluation trajectories(experimental setup in Section 5.1) exposes this gap. Figure 1(a) shows that 49.1% of logged eight-trajectory groups have homogeneous terminal outcomes, including 25.8% all-fail groups. Among all-fail evaluation groups involving tools, 57.4% contain trajectories with different numbers of completed tool-feedback cycles (Figure 1(b)), while 82.8% of failed trajectories involving tool feedback complete at least two cycles (Figure 1(c)). **Terminal rewards therefore conceal differences both across failed trajectories and among decisions within a trajectory**, motivating targeted alternative continuations from already reached states.

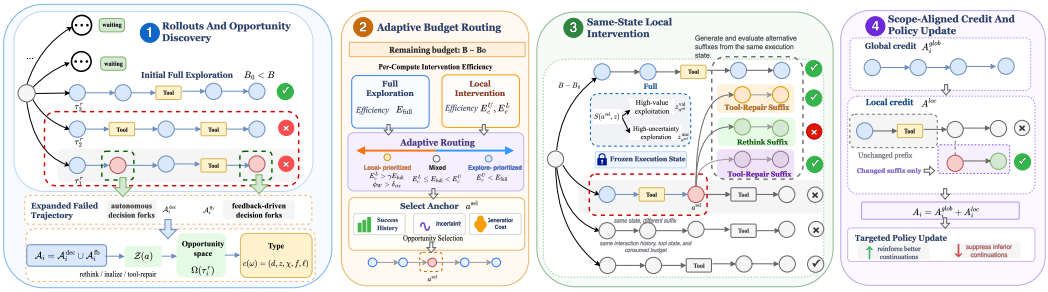


Figure 2: Overview of CALIPER.

We introduce **CALIPER**, which converts decisions reached in failed trajectories into local intervention opportunities under a fixed rollout-slot budget. CALIPER first allocates part of the budget to complete-trajectory exploration, which both searches for successful solutions and exposes reusable decision states when trajectories fail. From these failures, it identifies two types of forks: autonomous decision forks within model-generated reasoning or actions, and feedback-driven forks where the model must adjust its behavior after receiving new tool observations. Each fork is paired with feasible rethink, finalize, or tool-repair interventions. Using recent intervention success rates, uncertainty, and additional generation cost, CALIPER estimates the value of these opportunities relative to full exploration and dynamically allocates the remaining rollout slots between new complete trajectories and local continuations.

For a selected fork, CALIPER restores the same recorded interaction context, including prior tool outputs and consumed tool and interaction budgets, and resamples only the subsequent suffix. The original failed continuation and the alternative suffixes therefore share an identical pre-fork state, enabling verifiable same-state comparisons whose outcome differences can be attributed to post-fork behavior. During training, CALIPER combines global trajectory advantages with reliability-calibrated local Leave-One-Out (LOO) credit and restricts the latter to tokens generated after the fork. As a result, even when all full trajectories fail and provide no relative terminal signal, a successful local retry can still yield non-zero localized credit. When strict outcomes alone cannot distinguish failed continuations, bounded failure-quality evidence provides auxiliary supervision,

while targeted negative updates strengthen suppression only for post-fork behavior identified as locally inferior.

With eight rollout slots per prompt, CALIPER (Qwen3-8B) achieves an average score of 71.38 across five benchmarks, improving over Plain GRPO by 2.94 points and attaining the highest average among compared methods. It improves over Plain GRPO on all five benchmarks, shows the same overall trend with Qwen3-4B, and yields larger gains on GPQA-Diamond (Rein et al., 2024), LiveCodeBench (Jain et al., 2025), and more tool-intensive trajectories. Removing any component reduces average performance.

Our contributions:**1)**We introduce CALIPER, which turns autonomous and feedback-driven decisions in failed trajectories into assessable local intervention opportunities and adaptively coordinates their sampling with full exploration. **2)**We construct execution-consistent same-state comparisons and confine strict and auxiliary local credit, together with targeted negative updates, to post-fork behavior. **3)**Under matched rollout budgets, CALIPER improves performance across five benchmarks and two model scales, with analyses characterizing when local intervention is most useful.

2 Related Work

GRPO uses group-relative terminal rewards, while ToRL (Li et al., 2025) and Search-R1 (Jin et al., 2025) extend outcome-based RL to multi-turn tool use. Such trajectory-level signals cannot resolve local differences among failures with identical outcomes. CALIPER instead reuses reached states in failed trajectories to construct matched continuations, recovering local supervision without regenerating shared prefixes or increasing the rollout budget. Intermediate-state methods increasingly combine branching with adaptive allocation (Hou et al., 2025; Zhang et al., 2024; Li et al., 2026; Zong et al., 2026). VinePPO (Kazemnejad et al., 2025) and Tree-GRPO (Ji et al., 2025) resample or expand intermediate prefixes; ARPO (Dong et al., 2025b) and AEPO (Dong et al., 2025a) use entropy for branch selection and allocation; TRACE Zou et al. (2026), InfoTree (Hu et al., 2026), and CBPO (Wang et al., 2026) further incorporate value, uncertainty, coverage, or generation cost. CALIPER instead directly compares local interventions with fresh full trajectories under the same fixed budget, routing computation using recent success, uncertainty, and additional generation cost. Fine-grained methods derive local credit from alternative behaviors (Wei et al., 2026; Chang et al., 2026; Luo et al., 2025): GiGPO (Feng et al., 2025) compares same-state actions, ELPO (Liang et al., 2026) resamples suffixes around irrecoverable decisions, and CBPO exploits reward variation among exact-prefix continuations. CALIPER anchors both sampling and learning at the restored execution state, assigns signed Leave-One-Out credit only after the fork, and strengthens negative updates only for locally inferior continuations (Lai et al., 2024; Hwang et al., 2024; Wang et al., 2024; Cui et al., 2025).

3 Preliminary

3.1 Problem Formulation

We consider tool-augmented multi-step reasoning. Given a task $x \in \mathcal{X}$ and tool set $\mathcal{U}$, an agent follows policy π_θ. At interaction step $q = 1, \ldots, Q$, the state $s_q = (x, \mathcal{U}, h_q)$ contains the task, available tools, and prior model outputs and tool feedback h_q. The agent samples $u_q \sim \pi_\theta(\cdot \mid s_q)$ to continue reasoning, answer, or invoke a tool, whose observation updates the next state. A complete interaction forms $\tau = ((s_1, u_1), \ldots, (s_Q, u_Q))$. Strict supervision is available only at termination, with task-specific verifier V_x defining $y(\tau) = \mathbb{I}[V_x(\tau) = \text{success}] \in \{0, 1\}$. Here, $\mathbb{I}[\cdot]$ is the indicator function. For mathematical and scientific tasks, V_x checks the final answer; for coding tasks, it requires the generated program to pass all tests. Thus, the entire interaction is summarized by a binary terminal outcome.

3.2 Group-Based Agentic Reinforcement Learning

For each task x, the old policy $\pi_{\theta_{\text{old}}}$ samples G complete trajectories $\mathcal{G} = \{\tau_i\}_{i=1}^{G}$ with rewards $y_i = y(\tau_i)$. Group-relative optimization assigns each trajectory

$$A_i^{\mathrm{grp}} = \frac{y_i - \bar{y}_{\mathcal{G}}}{\sigma_{\mathcal{G}} + \epsilon_{\mathrm{num}}}. \tag{1}$$

Here, $\bar{y}_{\mathcal{G}}$ and $\sigma_{\mathcal{G}}$ are the group reward mean and sample standard deviation, $\epsilon_{\mathrm{num}} > 0$ ensures numerical stability, and $A_i^{\mathrm{grp}} = 0$ when all group outcomes are identical. For policy token $w_{i,t}$ with generation context $c_{i,t}$, let $M_{i,t} \in \{0, 1\}$ denote the training mask, with $M_{i,t} = 1$ only for policy-generated tokens. The importance ratio is $\rho_{i,t}(\theta) = \pi_\theta(w_{i,t} \mid c_{i,t})/\pi_{\theta_{\mathrm{old}}}(w_{i,t} \mid c_{i,t})$. The PPO clipped objective (Schulman et al., 2017) is

$$L_{\mathrm{clip}}(\theta, i, t) = \min\left(\rho_{i,t} A_i^{\mathrm{grp}}, \mathrm{clip}\left(\rho_{i,t}, 1 - \epsilon_l, 1 + \epsilon_h\right) A_i^{\mathrm{grp}}\right). \tag{2}$$

Here, ϵ_l and ϵ_h are the lower and upper clipping coefficients. Training maximizes its token-level average:

$$\max_\theta \ \mathbb{E}\left[\frac{\sum_{i=1}^{G}\sum_t M_{i,t} L_{\mathrm{clip}}(\theta, i, t)}{\sum_{i=1}^{G}\sum_t M_{i,t}}\right]. \tag{3}$$

Using complete trajectories as the common unit of sampling, comparison, and optimization creates three granularity mismatches: reached failure states are difficult to reuse, terminal rewards cannot resolve local behavioral differences, and trajectory-level credit propagates to decisions unrelated to the outcome difference. CALIPER instead refines training along three dimensions: *local opportunity construction, compute allocation, and credit scope*.

4 METHOD

CALIPER no longer treats a complete trajectory as the only training unit. Instead, it converts decision forks in failed trajectories and their feasible interventions into local intervention opportunities. At this intermediate granularity, CALIPER allocates rollout computation between full exploration and local retries, compares alternative continuations from matched execution states, and assigns local credit only to behavior changed by the intervention.

4.1 LOCAL INTERVENTION OPPORTUNITY SPACE

Initial Exploration and Decision Forks. Given a task x and rollout budget B, CALIPER first uses $B_0 < B$ slots for independent full-trajectory exploration, $\mathcal{R}_x^{(1)} = \{\tau_i^r\}_{i=1}^{B_0}$, and constructs local opportunities only from failed trajectories with $y(\tau_i^r) = 0$. Successful trajectories already provide positive supervision, whereas failures expose deficient decisions while preserving states reached with prior computation. For each failed trajectory τ_i^r, CALIPER identifies decision forks $\mathcal{A}_i = \mathcal{A}_i^{\mathrm{dec}} \cup \mathcal{A}_i^{\mathrm{fb}}$. Each fork a is associated with a recorded interaction state and token boundary k_a from which the suffix can be regenerated while preserving the preceding context. Here, $\mathcal{A}_i^{\mathrm{dec}}$ contains autonomous forks in model-generated reasoning or action text, while $\mathcal{A}_i^{\mathrm{fb}}$ contains feedback-driven forks after tool returns. Autonomous forks are restricted to resumable closed sentence, formula, or code-block boundaries, whereas feedback-driven forks are anchored at tool-return boundaries. For a candidate fork a with local window $\mathcal{W}_a$, CALIPER measures uncertainty as $U(a) = |\mathcal{W}_a|^{-1} \sum_{t \in \mathcal{W}_a} \mathsf{H}(\pi_\theta(\cdot \mid c_{i,t}))$ and retains high-$U(a)$ valid forks within separate relative-position intervals to balance uncertainty and trajectory coverage.

Constructing Local Intervention Opportunities. For each fork a, CALIPER considers three interventions and removes those infeasible under the remaining generation, interaction, or tool budget. *Rethink* resamples from the fork; *finalize* requests a final answer without further tool calls; and *tool-repair* is available only at tool-return forks classified as tool/runtime failures with sufficient remaining budget, allowing one new non-duplicate tool call before finalization. These interventions constrain continuation behavior without inserting new instructions into the recorded context. A feasible pair $\omega = (a, z)$, where $z \in \mathcal{Z}(a)$, defines a local intervention opportunity. For a failed trajectory, $\Omega(\tau_i^r) = \{(a, z) \mid a \in \mathcal{A}_i,\ z \in \mathcal{Z}(a)\}$, and CALIPER pools them as $\Omega_x = \bigcup_{i:y(\tau_i^r)=0} \Omega(\tau_i^r)$.

Each opportunity is typed by $c(\omega) = (d, z, \chi, f, \ell)$, representing task domain, intervention type, decision-fork type, failure type, and coarse relative position, respectively, so that evidence can be shared across similar interventions.

4.2 Compute-Aware Opportunity Routing

Local interventions reuse generated prefixes but have uncertain effectiveness, whereas full trajectories provide broader exploration at higher generation cost. CALIPER therefore routes the remaining rollout budget using strict success per additional generated token as a proxy for computation.

Per-Compute Intervention Efficiency. Because intervention effectiveness changes with the policy, CALIPER maintains statistics over the latest W training steps and backs off sparsely observed fine-grained types c to coarser types. For type c, let n_c and s_c be its intervention and strict-success counts. Under a prior $p_c \sim \mathrm{Beta}(\alpha_0, \beta_0)$, the posterior is $p_c \sim \mathrm{Beta}(\alpha_0 + s_c, \beta_0 + n_c - s_c)$, with mean $\mu_c = (s_c + \alpha_0)/(n_c + \alpha_0 + \beta_0)$ and variance $\sigma_c^2 = \mu_c(1-\mu_c)/(n_c + \alpha_0 + \beta_0 + 1)$. Let $\bar{T}_c$ denote average additional generation length; when no observations exist, it is initialized from the candidate fork's remaining generation budget and intervention-specific limit. With uncertainty adjustments $\kappa_U, \kappa_L \geq 0$, CALIPER defines

$$E_c^U = \frac{\min(1, \mu_c + \kappa_U \sigma_c)}{\bar{T}_c}, \qquad E_c^L = \frac{\max(0, \mu_c - \kappa_L \sigma_c)}{\bar{T}_c}. \tag{4}$$

For full exploration, μ_{full} and $\bar{T}_{\text{full}}$ are the smoothed strict-success rate and average generation length estimated from recent history and the current initial full trajectories, giving $E_{\text{full}} = \mu_{\text{full}}/\bar{T}_{\text{full}}$.

Opportunity Selection and Adaptive Allocation. Each feasible fork-intervention pair is scored by

$$S(a, z) = \frac{\min\big(1, \mu_{c((a,z))} + \kappa_s \sigma_{c((a,z))}\big)}{\bar{T}_{c((a,z))}}, \tag{5}$$

where $\kappa_s \geq 0$ controls selection uncertainty separately from routing. CALIPER sets $V(a) = \max_{z \in \mathcal{Z}(a)} S(a, z)$, selects $a^{\text{sel}} = \arg\max_a V(a)$, and chooses $z_{a^{\text{sel}}}^{\text{val}} = \arg\max_z S(a^{\text{sel}}, z)$; ties favor forks with higher $U(a)$. The selected pair determines the opportunity used for routing, and all allocated local suffixes branch from this same anchor.

If $\Omega_x = \varnothing$, all remaining slots are assigned to full exploration. Otherwise, let $c^\star = c((a^{\text{sel}}, z_{a^{\text{sel}}}^{\text{val}}))$. With sparse type history, CALIPER permits globally capped local probes alongside full exploration. With sufficient evidence, all remaining slots go to full trajectories when $E_{c^\star}^U < E_{\text{full}}$. After warmup, all remaining slots are assigned to local continuations only when $E_{c^\star}^L > \gamma E_{\text{full}}$ and $\phi_W \geq \delta_{\text{ctr}}$; otherwise CALIPER uses limited local probes together with full exploration, including the uncertain regime $E_{c^\star}^L \leq E_{\text{full}} < E_{c^\star}^U$. Here, $\gamma > 0$ is the efficiency margin, $\delta_{\text{ctr}} \in [0, 1]$ is the contrast threshold, and ϕ_W is the recent fraction of local-probe groups exhibiting either strict terminal-outcome or auxiliary failure-quality contrast between the pivot and suffixes, measured before auxiliary-credit gating. Initial full trajectories contribute to the current full-exploration estimate, whereas local outcomes enter intervention history only after allocation. Exact slot counts are given in the implementation details.

4.3 Execution-Consistent Local Interventions

Execution-State Preservation. Valid local comparison requires the original and resampled continuations to share the same pre-fork execution state; otherwise, outcome differences may arise from different tool observations or remaining budgets. For the selected fork a^{sel}, CALIPER restores its recorded context $e_{a^{\text{sel}}}$, including prior tool outputs, image inputs when present, prior tool-call signatures, and consumed tool and interaction budgets. The original pivot and m resampled suffixes:

$$\tau^p = (e_{a^{\text{sel}}}, \xi^p_{>a^{\text{sel}}}), \qquad \tau^s_j = (e_{a^{\text{sel}}}, \tilde{\xi}^{(j)}_{>a^{\text{sel}}}), \quad j = 1, \ldots, m. \tag{6}$$

Each new continuation follows its assigned intervention $z_j \in \mathcal{Z}(a^{\text{sel}})$ while reserving sufficient budget to reach a verifier-evaluable terminal state. The copied prefix only reconstructs $e_{a^{\text{sel}}}$ and is masked from suffix training.

Adaptive Intervention Exploration. For the selected fork, let $\mathcal{Z}_{a^{\text{sel}}} = \mathcal{Z}(a^{\text{sel}})$ and $c_{a^{\text{sel}},z} = c((a^{\text{sel}}, z))$. The value-oriented intervention is $z^{\text{val}}_{a^{\text{sel}}} = \arg\max_{z \in \mathcal{Z}_{a^{\text{sel}}}} S(a^{\text{sel}}, z)$, while the most uncertain alternative is $z^{\text{unc}}_{a^{\text{sel}}} = \arg\max_{z \in \mathcal{Z}_{a^{\text{sel}}} \setminus \{z^{\text{val}}_{a^{\text{sel}}}\}} \sigma_{c_{a^{\text{sel}},z}}$. If no alternative exists, CALIPER sets $z^{\text{unc}}_{a^{\text{sel}}} = z^{\text{val}}_{a^{\text{sel}}}$, and ties favor less-explored interventions. In the limited-probe regime, the value- and uncertainty-oriented interventions are each sampled once; when evidence favors local sampling, each is sampled twice. CALIPER therefore uses at most two intervention types, always from the same anchor, while online statistics balance high-value exploitation and high-uncertainty exploration.

4.4 Dual-Scale Credit Modeling and Targeted Updates

Full trajectories provide task-level supervision, whereas matched local continuations expose decision-level differences. CALIPER combines both scales, using historical evidence only to calibrate local reliability while current outcomes determine the direction of the update.

Dual-Scale Strict Credit. Let $\mathcal{R}_x$ denote all routed full-exploration trajectories. Their global advantage is $A^{\text{glob}}_i = (y_i - \bar{y}_{\mathcal{R}})/(\sigma_{\mathcal{R}} + \epsilon_{\text{num}})$ with token-level credit $A^{\text{glob}}_{i,t} = A^{\text{glob}}_i M_{i,t}$; if all outcomes are identical, $A^{\text{glob}}_i = 0$. For a selected fork a, the pivot and m suffixes form the same-state group $\mathcal{G}_a = \{\tau^p, \tau^s_1, \ldots, \tau^s_m\}$. CALIPER defines the Leave-One-Out baseline and local contrast as

$$B^{\text{LOO}}_i = \frac{\sum_{j \neq i,\, \tau_j \in \mathcal{G}_a} y_j}{|\mathcal{G}_a| - 1}, \qquad \hat{A}^{\text{loc}}_i = \text{clip}\left(y_i - B^{\text{LOO}}_i, -1, 1\right). \tag{7}$$

Because local groups are small and rewards are binary, CALIPER uses direct outcome contrasts without variance normalization. Let n^{hist}_a be the historical count of the selected opportunity's backed-off type before the current group. Probe and historical reliability are $r^{\text{probe}}_a = m/(m + \eta_p)$ and $r^{\text{hist}}_a = r_0 + (1 - r_0) n^{\text{hist}}_a/(n^{\text{hist}}_a + \eta_h)$, respectively. CALIPER then uses

$$\lambda_a = \min\left(\lambda_{\max}, \lambda_0 r^{\text{probe}}_a r^{\text{hist}}_a\right), \qquad A^{\text{loc}}_{i,t} = \lambda_a \hat{A}^{\text{loc}}_i M^{\text{loc}}_{i,t}, \quad M^{\text{loc}}_{i,t} = \mathbb{I}[t > k_a] M_{i,t}. \tag{8}$$

$\eta_p, \eta_h > 0$ control reliability saturation, $r_0 \in [0, 1]$ is the historical reliability floor, and $\lambda_0, \lambda_{\max} > 0$ bound local-credit strength. Thus, local credit applies only after the fork, and a successful local suffix can yield non-zero strict credit even when all full trajectories fail and $A^{\text{glob}} = 0$.

Bounded Failure-Quality Shaping and Credit Coordination. Strict rewards cannot distinguish failed samples, whereas unrestricted process signals may deviate from the task objective. CALIPER therefore uses auxiliary quality only among failures. It defines $q_i = \sum_{g \in \mathcal{Q}} \alpha_g q^{(g)}_i$ with $\mathcal{Q} = \{\text{valid}, \text{prog}, \text{uncens}, \text{nondup}\}$ and $\sum_g \alpha_g = 1$, measuring output validity, verifiable partial progress, non-truncation, and non-duplicative tool behavior. Progress is used only when verifiable. For any failed comparison set $\mathcal{F}$, define $\Delta q^{\mathcal{F}}_i = q_i - \sum_{j \neq i, \tau_j \in \mathcal{F}} q_j/(|\mathcal{F}| - 1)$. Using $\mathcal{F}_x = \{\tau^r_i \in \mathcal{R}_x \mid y_i = 0\}$ and $\mathcal{F}_a = \{\tau_i \in \mathcal{G}_a \mid y_i = 0\}$, the unscaled auxiliary credits are $\tilde{A}^{\text{glob,aux}}_{i,t} = \lambda_{\text{aux}} \Delta q^{\mathcal{F}_x}_i M_{i,t}$ and $\tilde{A}^{\text{loc,aux}}_{i,t} = \lambda_{\text{aux}} \Delta q^{\mathcal{F}_a}_i M^{\text{loc}}_{i,t}$, with zero auxiliary advantage for successful samples or invalid LOO comparisons.

To prevent auxiliary signals from overriding strict supervision, let $\Gamma_{\text{strict}} = \|A^{\text{glob}}\|_1$ and $\Gamma_{\text{aux}} = \|\tilde{A}^{\text{glob,aux}}\|_1 + \|\tilde{A}^{\text{loc,aux}}\|_1$. Given $\rho_{\text{aux}} \in (0, 1)$, CALIPER uses

$$\beta_{\text{aux}} = \mathbb{I}[\Gamma_{\text{strict}} > 0] \min\left(1, \frac{\rho_{\text{aux}}}{1 - \rho_{\text{aux}}} \frac{\Gamma_{\text{strict}}}{\Gamma_{\text{aux}} + \epsilon_{\text{num}}}\right), \tag{9}$$

and sets $A^{\text{glob,aux}} = \beta_{\text{aux}}\tilde{A}^{\text{glob,aux}}$ and $A^{\text{loc,aux}} = \beta_{\text{aux}}\tilde{A}^{\text{loc,aux}}$. CALIPER further coordinates global and local credit using $\Gamma_{\text{global}} = \|A^{\text{glob}}\|_1 + \|A^{\text{glob,aux}}\|_1$ and $\Gamma_{\text{local}} = \|A^{\text{loc}}\|_1 + \|A^{\text{loc,aux}}\|_1$. For $\rho_{\text{loc}} \in (0,1)$, it sets $\beta_{\text{loc}} = \min(1, \frac{\rho_{\text{loc}}}{1-\rho_{\text{loc}}} \frac{\Gamma_{\text{global}}}{\Gamma_{\text{local}}+\epsilon_{\text{num}}})$ when $\Gamma_{\text{global}} > 0$, and $\beta_{\text{loc}} = 1$ otherwise. The final local and token-level credits are

$$\bar{A}^{\text{loc}}_{i,t} = \beta_{\text{loc}} \left(A^{\text{loc}}_{i,t} + A^{\text{loc,aux}}_{i,t} \right), \qquad A_{i,t} = A^{\text{glob}}_{i,t} + A^{\text{glob,aux}}_{i,t} + \bar{A}^{\text{loc}}_{i,t}. \tag{10}$$

Thus, task-level and decision-level supervision contribute jointly while their magnitudes remain controlled.

Local-Contrast-Driven Targeted Updates. Same-state comparisons identify locally inferior continuations. CALIPER strengthens negative updates only when both local credit and the final advantage are negative, using $M^{-}_{i,t} = \mathbb{I}[\bar{A}^{\text{loc}}_{i,t} < 0]\mathbb{I}[A_{i,t} < 0]M^{\text{loc}}_{i,t}$. It sets $\epsilon^{-}_{i,t} = \epsilon^{\text{loc}}_{l}$ when $M^{-}_{i,t} = 1$ and $\epsilon^{-}_{i,t} = \epsilon_l$ otherwise, where $\epsilon^{\text{loc}}_{l} > \epsilon_l$. In the clipped objective from the Preliminary, $1 - \epsilon_l$ is therefore replaced by $1 - \epsilon^{-}_{i,t}$ while the upper bound remains unchanged. Using the same token-level aggregation, stronger suppression is confined to post-fork continuations identified as inferior by local comparison, while all other regions retain standard PPO updates.

5 Experiments

5.1 Experimental Setup

Training Data and Evaluation. To isolate gains from reinforcement learning, we train directly from Qwen3-8B (Yang et al., 2025) without task-specific SFT. We use a fixed source-stratified subset of 8,320 Open-AgentRL-30K (Yu et al., 2025b) examples, preserving the original source proportions, and train for 130 optimizer steps with 64 prompts per step, traversing the subset once. We evaluate on AIME 2025 (Zhang & Math-AI, 2025), MATH-500, GSM8K, GPQA-Diamond, and LiveCodeBench v6 under a unified multi-turn TIR setting, reporting pass@8/2/2/4/2, respectively. Full data composition and evaluation protocols are provided in Appendix A.3.

Training Protocol. Each prompt has a fixed rollout budget of $B = 8$, with $B_0 = 4$ slots used for initial full-trajectory exploration. CALIPER dynamically allocates the remaining slots between full trajectories and local suffixes, yielding root/suffix allocations of $8/0$, $6/2$, or $4/4$. These allocation levels are calibrated in advance using a held-out split and then fixed for all main experiments; details are provided in Appendix A.4.3. All methods use the same rollout-slot budget, with auxiliary sampling counted toward it. Additional training details are provided in Appendix A.2.

Baselines. We compare CALIPER with Qwen3-8B, Plain GRPO, and representative agentic RL methods covering trajectory-level optimization, adaptive rollout allocation, and fine-grained credit assignment: DAPO (Yu et al., 2025a), ARPO, AEPO, GiGPO, ELPO, and VinePPO. All re-trained methods follow the same experimental protocol while preserving their core algorithms; implementation details appear in Appendix A.5.

5.2 Main Results

Best Overall Performance. Under the same rollout-slot budget, CALIPER achieves the highest average score of **71.38** with Qwen3-8B, exceeding Plain GRPO by **2.94** points. It improves over Plain GRPO on all five benchmarks, and the Qwen3-4B results reproduce the same overall trend. These results show that jointly improving sampling, comparison, and credit assignment makes more effective use of a fixed rollout budget.

Larger Gains on More Challenging Tasks. CALIPER obtains its largest improvements over Plain GRPO on GPQA (**+5.55**), LiveCodeBench (**+3.43**), and AIME25 (**+3.34**), while also improving MATH500 (Lightman et al., 2023) and GSM8K (Cobbe et al., 2021). These tasks involve longer-horizon reasoning, code generation, or richer tool interaction, where terminal rewards provide limited information about which intermediate decisions determine the outcome. The pattern is

Table 1: Main results on five benchmarks under the unified multi-turn TIR setting.

Method	AIME25	MATH500	GSM8K	GPQA	LiveCodeBench	Avg.
Qwen3-8B	66.67	88.40	92.27	59.60	28.57	67.10
Plain GRPO	73.33	88.60	92.65	59.60	28.00	68.44
GiGPO	70.00	89.40	93.18	62.12	28.57	68.65
AEPO	74.44	89.20	92.87	61.62	28.57	69.34
ARPO	71.11	89.00	93.18	61.11	30.29	68.94
ELPO	73.33	87.60	**94.16**	63.64	28.00	69.35
DAPO	73.33	89.40	93.25	58.59	30.29	68.97
VinePPO	**76.67**	89.80	93.18	61.62	29.14	70.08
CALIPER (ours)	**76.67**	**90.00**	93.63	**65.15**	**31.43**	**71.38**
Qwen3-4B	70.00	90.00	92.57	**60.10**	28.57	68.25
Plain GRPO (4B)	73.33	89.60	93.18	58.59	28.57	68.65
CALIPER (4B)	**76.67**	**90.80**	**95.00**	59.60	**29.14**	**70.24**

consistent with CALIPER's use of prioritized local interventions, execution-consistent comparisons, and post-fork credit.

Consistent Performance Across Tasks. CALIPER achieves or ties the best result on four of five benchmarks and remains close to the best on GSM8K. Its gains span mathematical reasoning, scientific reasoning, and code generation, indicating consistent improvements across task and capability types.

Table 2: Ablation study of CALIPER.

Method	AIME25	MATH500	GSM8K	GPQA	LCB	Avg.
CALIPER	**76.67**	**90.00**	**93.63**	**65.15**	**31.43**	**71.38**
w/o Adaptive Routing	70.00	89.40	93.03	60.61	28.57	68.32
w/o Feedback-driven Forks	66.67	89.00	93.18	63.64	29.14	68.33
w/o Matched Local Credit	66.67	69.00	92.87	62.63	29.71	64.18
w/o Opportunity Prioritization	73.33	88.80	93.32	64.65	28.57	69.73
w/o Scope Alignment	60.00	89.20	92.80	60.61	29.14	66.35
w/o Targeted Negative Update	73.33	87.80	92.49	61.11	30.28	69.00

5.3 Ablation Study

Removing any CALIPER component lowers average performance. The largest drops occur when removing matched local credit, with the average score decreasing from 71.38 to 64.18, and scope alignment, with the score decreasing to 66.35. Removing adaptive routing, feedback-driven forks, opportunity prioritization, or targeted negative updates also degrades performance. These results indicate that the components make complementary contributions to the full method.

5.4 Analysis

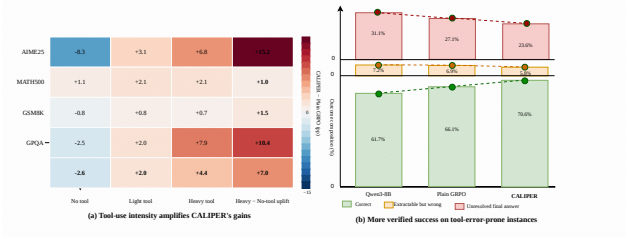


Figure 3: CALIPER Shows Larger Gains with Increasing Tool Interaction.

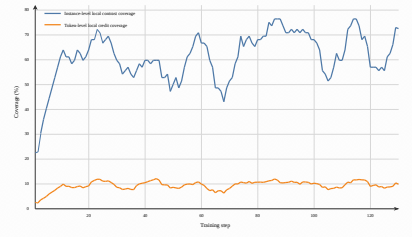


Figure 4: Scope Alignment Localizes Local Credit.

CALIPER's gains amplify with increasing tool interaction. Using Qwen3-8B rollouts to define instance properties, we group evaluation instances by average tool calls into no-tool, light (≤ 2), and heavy (> 2) usage. As shown in Figure 3(a), CALIPER's macro-average correctness gain over Plain GRPO rises from -2.6 points to $+2.0$ and $+4.4$ points across these groups. More tool interaction introduces additional tool-return boundaries and feedback-conditioned decisions, where CALIPER's feedback-driven forks become increasingly relevant. Among instances where the base model encounters tool-execution errors, CALIPER also raises successful final outcomes from 66.1% to 70.6% while reducing erroneous and unresolved outcomes (Figure 3(b)), indicating more reliable use of tool feedback.

Scope alignment localizes intervention-derived credit. Across 130 Qwen3-8B training steps, valid local comparisons are formed for 61.1% of task instances on average, yet only 9.4% of response tokens receive local credit(Figure 4). Activated regions consistently coincide with post-fork intervention intervals, while reused prefixes are never optimized. Thus, CALIPER provides additional local supervision to many training instances while restricting its effect to behavior actually changed by the intervention, avoiding credit propagation through shared history.

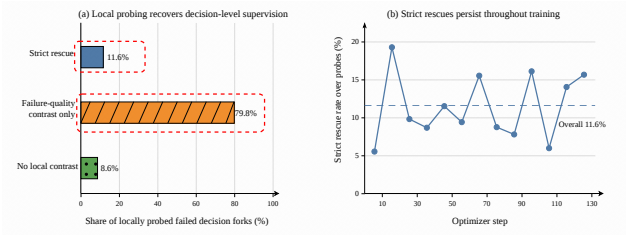


Figure 5: Local Interventions Recover Signals Hidden by Terminal Failures.

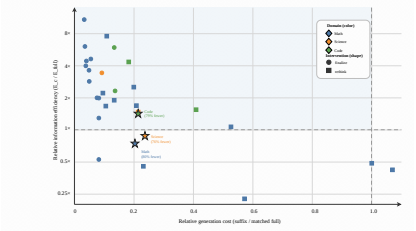


Figure 6: Local Opportunities Vary in Compute Efficiency.

CALIPER increases training-signal density under sparse terminal supervision. We analyze the failed decision forks that are actually selected for local probing during training, and we categorize outcomes as **strict rescue** when a failed pivot yields at least one successful suffix, **failure-quality contrast** when all continuations fail but differ in validity, verifiable progress, non-truncation, or non-duplicative tool behavior, and **no local contrast** otherwise. Figure 5(a) shows that 11.6% achieve strict rescue and 79.8% yield failure-quality contrast, leaving only 8.6% without local distinction. Thus, 91.4% of selected forks reveal information absent from their original terminal outcomes. Strict rescues occur throughout training (Figure 5(b)), providing verified local credit, while failure-quality contrasts supply gated auxiliary evidence.

Local opportunities exhibit heterogeneous compute efficiency. We analyze opportunity types observed at least 8 times during the final 32 training steps. Figure 6 compares each type with full rollouts using newly generated suffix tokens and success per generated token. Local suffixes require only about 20% of the newly generated tokens of full rollouts on average, reducing generation cost by 79.3%, and 82.14% of opportunity types are more compute-efficient than full exploration. However, some remain below $1\times$ efficiency, demonstrating substantial heterogeneity and motivating CALIPER's routing based on historical success, uncertainty, and generation cost.

6 Conclusion

We study how to recover decision-level supervision from failed tool-augmented trajectories under a fixed rollout budget. CALIPER adaptively combines full exploration with same-state local retries and confines retry-derived credit to post-fork behavior. Under matched budgets, CALIPER achieves an average score of 71.38 with Qwen3-8B, improving over Plain GRPO by 2.94 points, with consistent gains on Qwen3-4B. Overall, coordinating sampling, comparison, and credit assignment improves training-signal utilization under limited rollout computation.

AI USE STATEMENT

Generative AI tools were used only for minor language editing during manuscript preparation, such as correcting grammatical errors, adjusting sentence structure and word order, and improving linguistic fluency. They were not used to generate research ideas, technical content, methodology, experimental designs, results, analyses, or scientific conclusions. All substantive content of the paper was produced by the authors. The authors reviewed all AI-assisted edits and take full responsibility for the final manuscript.

ETHICS STATEMENT

This work focuses on methodological research on post-training tool-use agents. It does not involve human subjects, personally identifiable or sensitive information, or data collection from individuals. The experiments use existing research datasets, models, tools, and benchmarks in accordance with their intended research purposes and applicable licenses. The proposed method is evaluated only in research settings and does not involve deployment in high-stakes or safety-critical applications. We are not aware of any additional ethical concerns specific to this work.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Qikai Chang, Zhenrong Zhang, Pengfei Hu, Jun Du, Jiefeng Ma, Yicheng Pan, Jianshu Zhang, Quan Liu, and Gao Jianqing. Thor: Tool-integrated hierarchical optimization via rl for mathematical reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 21487–21510, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/245b7f8357fdc26797ed6714cda5bd94-Paper-Conference.pdf.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, ZhouYijie, Chenzheng Zhu, Haofen Wang, Jeff Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and weipeng chen. Research: Learning to reason with search for llms via reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 85287–85307. Curran Associates, Inc., 2025. doi: 10.52202/085713-2858. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/7b3a0d3a0864b66f229c0a9c84f9e950-Paper-Conference.pdf.

Mingyue Cheng, Shuo Yu, Daoyu Wang, Qingchuan Li, Xiaoyu Tao, Jie Ouyang, Yucong Luo, Yitong Zhou, Qi Liu, and Enhong Chen. Agent-r1: A unified and modular framework for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2511.14460.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

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, 2025. URL https://arxiv.org/abs/2502.01456.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng

Dou. Agentic entropy-balanced policy optimization, 2025a. URL https://arxiv.org/abs/2510.14545.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025b. URL https://arxiv.org/abs/2507.19849.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 37909–37926, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4038c9208dfc22644c60ad39c24e5c53-Paper-Conference.pdf.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025. URL https://arxiv.org/abs/2505.10978.

Weiyang Guo, Zesheng Shi, Longhui Zhang, Zeen Zhu, Min Zhang, and Jing Li. Agent reinforcement learning via pivotal-aware self-feedback retry, 2026. URL https://arxiv.org/abs/2607.03702.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12355–12369, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.604. URL https://aclanthology.org/2025.acl-long.604/.

Yuelin Hu, Zhenbo Yu, Zhengxue Cheng, Wei Liu, and Li Song. Maximizing rollout informativeness under a fixed budget: A submodular view of tree search for tool-use agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2605.05262.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. Self-explore: Enhancing mathematical reasoning in language models with fine-grained rewards. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1444–1466, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.78. URL https://aclanthology.org/2024.findings-emnlp.78/.

Naman Jain, Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/94074dd5a072d28ff75a76dabed43767-Paper-Conference.pdf.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

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 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. 29557–29590. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms, 2024. URL https://arxiv.org/abs/2406.18629.

Kun Li, Zenan Xu, Junan Li, Zengrui Jin, Jinghao Deng, Zexuan Qiu, and Bo Zhou. Discovery and reinforcement of tool-integrated reasoning chains via rollout trees. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14265–14283, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.649. URL https://aclanthology.org/2026.acl-long.649/.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl, 2025. URL https://arxiv.org/abs/2503.23383.

Qiao Liang, Yuke Zhu, Chao Ge, Lei Yang, Ying Shen, Bo Zheng, and Sheng Guo. Learning from the irrecoverable: Error-localized policy optimization for tool-integrated LLM reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11008–11028, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.504. URL https://aclanthology.org/2026.acl-long.504/.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning, 2025. URL https://arxiv.org/abs/2508.03680.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Mengjie Ren, Jie Lou, Boxi Cao, Xueru Wen, Hongyu Lin, Xianpei Han, Le Sun, Xing Yu, and Yaojie Lu. Learning from failures: Correction-oriented policy optimization with verifiable rewards, 2026. URL https://arxiv.org/abs/2605.14539.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.05592.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Ying Wang, Changlin Qiu, Bang Lin, Linbo Jin, Wen Jiang, Zhe Sun, and Jingli Yang. Contrastive branch policy optimization, 2026. URL https://arxiv.org/abs/2608.24300.

Quan Wei, Siliang Zeng, Chenliang Li, Zhongruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via fine-grained reward structure and credit assignment, 2026. URL https://arxiv.org/abs/2505.11821.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yue Yu, 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*, 2025a. URL https://mlanthology.org/neurips/2025/yu2025neurips-dapo/.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025b.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. REST-MCTS*: LLM Self-Training via Process Reward Guided Tree Search. In *Neural Information Processing Systems*, 2024. doi: 10.52202/079017-2066. URL https://mlanthology.org/neurips/2024/zhang2024neurips-restmcts/.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Zefang Zong, Dingwei Chen, Yang Li, Qi Yi, Bo Zhou, Chengming Li, BO Qian, Peng Chen, and Jie Jiang. AT²PO: Agentic turn-based policy optimization via tree search. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24120–24143, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1106. URL https://aclanthology.org/2026.acl-long.1106/.

Heming Zou, Qi Wang, Yun Qu, Yuhang Jiang, Lizhou Cai, Yixiu Mao, Ru Peng, Xin Xu, Weijie Liu, Kai Yang, Saiyong Yang, and Xiangyang Ji. Trace: A unified rollout budget allocation framework for efficient agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2606.11119.

A Appendix

A.1 Limitations

CALIPER is designed for tool-augmented reinforcement learning with sparse terminal supervision, where failed trajectories expose intermediate execution states that can be restored and locally retried. Its benefit therefore depends on the existence of meaningful resumable states and verifiable continuation outcomes. Tasks with little sequential structure or few useful intermediate decision points may provide fewer opportunities for local intervention.

Not every failed trajectory necessarily contains a valuable local fork. Some failures may result from errors made substantially earlier than any recoverable boundary, while others may contain candidate

Table 3: Training configuration used for the main CALIPER experiments. The Qwen3-4B setting differs from Qwen3-8B only in model initialization.

Configuration	Value
Training examples	8,320
Optimizer steps	130
Prompts per step	64
Rollout slots per prompt	8
Initial full rollouts	4
PPO mini-batch size	16
PPO epochs	1
Actor learning rate	3×10^{-6}
LR schedule	Constant
Warmup ratio	0
Weight decay	0.01
Gradient clipping	1.0
PPO lower / upper clip	0.2 / 0.2
Entropy coefficient	0
KL reward coefficient	0
Actor KL-loss coefficient	0
Loss aggregation	Token mean
Maximum prompt length	2,560
Maximum response length	7,168
Training temperature	1.0
Training top-p	1.0
Training top-k	-1
Maximum tool calls	6
Assistant / user-response turns	7 / 6

forks whose alternative continuations remain uninformative. CALIPER mitigates this issue through opportunity prioritization and adaptive routing, but these mechanisms do not guarantee that every failed trajectory yields useful local supervision.

Our experiments cover Qwen3-4B and Qwen3-8B on five mathematical, scientific, and code-generation benchmarks under a Python-based multi-turn tool environment. The current study does not establish whether the same gains transfer unchanged to substantially larger models, other model families, browser or search agents, or open-ended environments without reliable task verifiers. Extending local intervention and verification to these settings remains an important direction for future work.

A.2 Training and Implementation Details

A.2.1 Model and Optimization

Our main experiments initialize directly from the post-trained Qwen3-8B checkpoint rather than Qwen3-8B-Base. We do not introduce an additional task-specific SFT stage before reinforcement learning.

The Qwen3-4B experiments replace the initialization with Qwen3-4B and otherwise use the same training configuration as the Qwen3-8B experiments, including the training data, number of optimizer steps, rollout budget, sampling configuration, CALIPER routing parameters, optimization hyperparameters, tool environment, and evaluation protocol.

For both model scales, we train LoRA adapters with rank $r = 64$, scaling parameter $\alpha = 32$, zero LoRA dropout, and no bias parameters. LoRA is applied to the attention `q/k/v/o_proj` modules and the MLP `gate/up/down_proj` modules.

Table 3 summarizes the training configuration shared by Qwen3-8B and Qwen3-4B unless otherwise stated.

Dynamic group filtering and overlength reward shaping are disabled. Training uses gradient checkpointing and padding removal.

A.2.2 Training Data and Prompt Construction

We use a fixed source-stratified subset of 8,320 examples from Open-AgentRL-30K, preserving the source proportions of the original corpus. Training runs for 130 optimizer steps with 64 prompts per step, traversing this subset exactly once.

Downstream evaluation benchmarks are not included in the training subset. The held-out data used for rollout-allocation calibration are also disjoint from both the 8,320-example training subset and all downstream evaluation benchmarks.

Training examples are processed according to their original data source. For `math_dapo` and `mega-science`, the original user prompt is retained and an output-format constraint is appended requiring the final numerical answer to appear inside `\boxed{}` without units.Other mathematical sources use a common tool-augmented reasoning instruction.

For code examples, prompt preprocessing removes a known unfinished `# YOUR CODE HERE` submission tail when present, closes unmatched code fences when necessary, and appends a single complete-Python-program submission instruction.

Each original example is provided as one user message without manually constructed few-shot demonstrations. The Qwen chat template subsequently inserts the system tool schema and assistant generation prompt. The maximum training prompt length is 2,560 tokens, with middle truncation for overlength inputs.

A.2.3 Hardware and Distributed Training

All reported Qwen3-8B and Qwen3-4B training runs use four NVIDIA A100 80GB GPUs.

Actor optimization uses FSDP with Ulysses sequence parallelism. Rollout generation uses asynchronous BF16 vLLM generation with tensor parallelism 1. Parameter and optimizer offloading are disabled.

A.2.4 Fixed Rollout-Slot Budget

For every training prompt, CALIPER operates under a fixed rollout budget of $B = 8$. The first $B_0 = 4$ slots always generate independent complete trajectories, which serve both as training samples and as probes of the current policy.

The remaining $R = B - B_0 = 4$ slots are allocated between additional full trajectories and conditional suffixes. CALIPER therefore uses one of three root/suffix allocation levels: $8/0$, $6/2$, or $4/4$.

A conditional suffix consumes one rollout slot even though it reuses an already generated prefix. Thus, local branching does not increase the total number of rollout slots available to CALIPER.

All auxiliary sampling is included in the same budget. In particular, value-oriented and uncertainty-oriented local continuations consume ordinary rollout slots rather than being generated outside the budget.

A.2.5 Training-Time Sampling

Training rollouts use stochastic decoding with $T = 1.0$, $p_{\text{top}} = 1.0$, top-k $= -1$, repetition penalty 1.0, and normal EOS termination.

Root trajectories follow the ordinary tool-agent policy and receive no CALIPER-specific textual intervention.

Conditional suffixes share the same 7,168-token total response budget as their corresponding root trajectory. To preserve sufficient capacity for a verifier-evaluable final answer, suffix interventions reserve 422 tokens for mathematical and scientific tasks and 768 tokens for code tasks. These reserved tokens are part of the original response budget rather than an additional allowance.

Table 4: Evaluation protocol. n denotes the number of independently sampled completions for each problem. All methods use the same Python tool environment and task-specific verifier.

Benchmark	Problems	n	Max response	Max turns	Max tool calls
AIME 2025	30	8	16,384	12 / 12	32
MATH-500	500	2	8,192	8 / 8	16
GSM8K	1,319	2	4,096	4 / 4	8
GPQA-Diamond	198	4	8,192	12 / 12	32
LiveCodeBench v6	175	2	16,384	12 / 12	32

A.2.6 Tool Environment

Training uses a Python `code_interpreter` backed by an isolated SandboxFusion environment. The tool schema contains a single string argument, `code`.

Each tool execution has a 30-second timeout and a 2,048 MB memory limit. The tool service uses 128 workers with a global rate limit of 128. At most one tool call is executed in parallel within an interaction turn.

Tool observations returned to the model are truncated to at most 256 characters while preserving information from both ends of an overlength output.

Each trajectory is allowed at most six tool calls, seven assistant turns, and six user/tool-response turns. Calling the tool does not itself increase the strict task reward.

For code-generation tasks, final program verification is performed separately by the reward scorer against task tests. This verification step does not consume the trajectory's interactive tool-call budget.

A.3 Evaluation Details

A.3.1 Benchmarks and Interaction Budgets

We evaluate all methods under the same multi-turn tool-integrated reasoning protocol. Evaluation uses the ordinary `tool_agent`; CALIPER's training-time branching, posterior updates, local-credit construction, and prefix reuse are disabled.

Table 4 summarizes the benchmark-specific settings.

AIME 2025. We use all 30 problems from AIME I and AIME II 2025. The final answer must appear in the last valid `\boxed{}` expression.

MATH-500. We use the 500-example test split of `HuggingFaceH4/MATH-500`. The final boxed answer must contain only the answer value and no unit.

GSM8K. We use all 1,319 examples from the `main/test` split. The reference answer is obtained from the value following the original `####` marker. The model must place only the final numerical value inside `\boxed{}`.

GPQA-Diamond. We evaluate on all 198 examples of the Diamond subset. The option ordering is fixed by the evaluation data, and the model is required to return the final uppercase option letter inside `\boxed{}`.

LiveCodeBench v6. We use 175 problems from the code-generation-lite v6 incremental set used by our evaluation pipeline. The problems span January 4 to April 6, 2025. Each model produces a complete Python program, which is subsequently evaluated using the pinned official public and private tests. The generation-side data provide the problem statement and problem identity; final correctness is determined by the official-test rescorer.

A.3.2 Evaluation Decoding

Evaluation uses stochastic generation with $T = 0.6$, $p_{\text{top}} = 0.95$, top-k $= -1$, repetition penalty 1.0, and normal EOS termination.

The maximum prompt length is 4,096 tokens, with middle truncation for overlength prompts. Evaluation uses BF16 asynchronous vLLM generation with tensor parallelism 4.

The effective top-k value is -1. Although an evaluation wrapper contains a `top_k=20` configuration entry, the asynchronous validation path used for the reported experiments does not propagate this value to the actual vLLM sampling request. We therefore report the value that is effective during generation.

Multiple completions of the same problem are independent stochastic generations under the same evaluation protocol; they are not treated as independent training seeds.

A.3.3 Answer Verification

Mathematical and Scientific Tasks. For AIME 2025, MATH-500, GSM8K, and GPQA-Diamond, the verifier searches the final 4,096 characters of the generated response and extracts the last complete brace-balanced `\boxed{...}` expression. A response without a valid boxed answer is scored as incorrect.

For AIME 2025, MATH-500, and GSM8K, verification proceeds through exact matching, normalized matching after removing benign formatting differences, and conservative mathematical-equivalence checking. A completion is correct if any stage verifies the extracted answer.

GPQA-Diamond uses the same boxed-answer extraction, with correctness determined by the final uppercase option letter.

Code Generation. LiveCodeBench is evaluated in two stages. The model first generates a complete agent trajectory. The final submitted program is then rescored using the pinned LiveCodeBench v6 official public and private tests.

If a response contains `</think>`, code extraction is applied to the subsequent text; otherwise the complete response is inspected. The scorer extracts the final complete lowercase `python` code fence.

A completion is correct only if the submitted program passes all official tests. Missing code fences, compilation failures, runtime failures, timeouts, and wrong answers are scored as incorrect. Sandbox infrastructure failures are treated as evaluation failures rather than silently converted into ordinary wrong answers.

Training rewards, auxiliary failure-quality scores, trajectory length, and tool-use statistics do not affect benchmark correctness.

A.3.4 Reported Metrics

The main paper reports $\text{pass@}8/2/2/4/2$ for AIME 2025, MATH-500, GSM8K, GPQA-Diamond, and LiveCodeBench v6, respectively.

For benchmark d, let Q_d denote the number of problems and let n_d be the prescribed number of independently sampled completions per problem. Let $y_{i,j} \in \{0, 1\}$ indicate whether completion j for problem i is correct. We compute

$$\text{pass@}n_d = \frac{1}{Q_d} \sum_{i=1}^{Q_d} \mathbb{I}\left[\sum_{j=1}^{n_d} y_{i,j} > 0\right]. \tag{11}$$

Thus, a problem is counted as solved when at least one of its n_d generated completions is correct. The aggregate score reported in Table 1 is the unweighted macro-average over the five benchmark scores,

$$\text{Avg.} = \frac{1}{5} \sum_{d=1}^{5} \text{pass@}n_d. \tag{12}$$

We do not weight this average by the number of problems or generated completions in each benchmark.

A.4 Additional CALIPER Implementation Details

A.4.1 Decision-Fork Extraction

CALIPER constructs local intervention opportunities only from unsuccessful initial full trajectories.

Autonomous Forks. For each sufficiently long model-generated segment, the implementation searches for resumable boundaries at a 16-token search stride. Candidate search begins after the first 64 generated tokens and leaves at least 64 tokens after the candidate location.

The search stride only proposes locations. A candidate is retained only if the decoded prefix ends at a structurally valid semantic boundary. Cuts inside unmatched tool-call tags, unmatched code fences, or unbalanced bracketed structures are rejected. Accepted boundary types include closed code blocks, balanced sentence or formula endings, and assistant-turn boundaries.

For a candidate ending at token k, uncertainty is measured using the mean token entropy over up to the preceding 64 generated tokens.

To avoid concentrating all candidates in one region, valid boundaries are ordered by position and divided into at most four contiguous groups. CALIPER retains the most uncertain valid boundary from each group, yielding at most four autonomous candidates from a generated segment.

Feedback-Driven Forks. Whenever an executed tool call returns an observation, the resulting post-tool state is stored as a feedback-driven candidate. The stored state includes the interaction history after the tool return, the already consumed tool and turn budgets, and the corresponding response position. Its uncertainty is measured from the model entropy associated with that interaction round.

Failure and Position Types. Failed root trajectories are assigned one of three coarse failure classes: `tool/runtime-error`, `no-final/length`, or `wrong-final`. Relative position is discretized into early, middle, and late buckets using boundaries at 0.34 and 0.67. Together with task domain, intervention type, and fork type, these attributes instantiate the opportunity type $c(\omega) = (d, z, \chi, f, \ell)$ defined in the main paper.

A.4.2 Intervention Feasibility

For each candidate fork, CALIPER considers three structural interventions: *rethink*, *finalize*, and, when feasible, *tool-repair*.

Rethink. The model resumes ordinary stochastic generation from the restored execution state.

Finalize. The model resumes from the same state with new tool calls disabled and is required to complete the final response.

Tool-Repair. Tool-repair is available only at a tool-return fork classified as a tool/runtime failure. It additionally requires sufficient remaining assistant-turn, user/tool-response, tool-call, and token budgets. The intervention allows one new non-duplicate tool call before finalization.

These interventions are implemented through structural generation constraints rather than by appending new natural-language instructions to the recorded conversation. Consequently, alternative continuations share exactly the context that the original trajectory reached.

Candidates that cannot preserve the required finalization token reserve are removed before opportunity selection.

A.4.3 CALIBRATION OF ROLLOUT ALLOCATION LEVELS

CALIPER uses a held-out, source-stratified calibration split from Open-AgentRL-30K to determine the discrete root/suffix allocation levels. The calibration data are disjoint from the 8,320-example training subset and all downstream evaluation benchmarks.

We use 1,024 prompts for calibration and apply the same rollout budget as in main training, with $B = 8$, $B_0 = 4$ initial full-trajectory slots, and $R = B - B_0 = 4$ remaining slots routed between fresh full trajectories and local continuations.

For each intervention type c, CALIPER maintains strict-success statistics over a sliding window of $W = 32$ training steps. We use a uniform Beta prior $p_c \sim \mathrm{Beta}(1, 1)$. Writing the posterior parameters as $\alpha_c = 1 + s_c$ and $\beta_c = 1 + n_c - s_c$, we obtain posterior mean μ_c and uncertainty σ_c. Let $\bar{T}_c$ denote the average number of newly generated tokens required by the intervention.

For allocation-level calibration, we use the cost-aware upper confidence bound

$$U_c(q) = \frac{\min(1, \mu_c + \kappa_{\mathrm{cal}}\sigma_c(q))}{\bar{T}_c}, \qquad \kappa_{\mathrm{cal}} = 1.5, \tag{13}$$

where q denotes the number of additional local samples already assigned virtually during allocation. We approximate the expected uncertainty after these samples by

$$\sigma_c^2(q) = \frac{\mu_c(1 - \mu_c)}{\alpha_c + \beta_c + q + 1}. \tag{14}$$

Thus, assigning additional local samples progressively reduces their uncertainty bonus. Fresh full-trajectory exploration is evaluated using its recent strict-success efficiency $E_{\mathrm{full}} = \mu_{\mathrm{full}}/\bar{T}_{\mathrm{full}}$.

At a selected decision fork, CALIPER considers at most two complementary local interventions: a value-oriented intervention z^{val} and an uncertainty-oriented intervention z^{unc}. One local probe batch samples each once and therefore consumes two rollout slots. Its marginal optimistic efficiency is

$$U_{\mathrm{loc}}^{(j)} = \frac{\sum_{z \in \{z^{\mathrm{val}}, z^{\mathrm{unc}}\}} \min\left(1, \mu_z + \kappa_{\mathrm{cal}}\sigma_z(q_z^{(j)})\right)}{\sum_{z \in \{z^{\mathrm{val}}, z^{\mathrm{unc}}\}} \bar{T}_z}. \tag{15}$$

Because only four slots remain after initial exploration, at most two local probe batches can be allocated. We allocate another batch only when its marginal cost-aware UCB remains no lower than the efficiency of fresh full exploration. This yields

$$\begin{cases} 8/0, & U_{\mathrm{loc}}^{(0)} < E_{\mathrm{full}}, \\ 6/2, & U_{\mathrm{loc}}^{(0)} \geq E_{\mathrm{full}} \text{ and } U_{\mathrm{loc}}^{(1)} < E_{\mathrm{full}}, \\ 4/4, & U_{\mathrm{loc}}^{(0)} \geq E_{\mathrm{full}} \text{ and } U_{\mathrm{loc}}^{(1)} \geq E_{\mathrm{full}}. \end{cases} \tag{16}$$

The calibration consistently supports these three allocation regimes, so we fix $8/0$, $6/2$, and $4/4$ before the main training runs. Downstream benchmark performance is not used to select or modify these levels. During main training, online statistics determine which calibrated regime is used for each prompt.

A.4.4 ONLINE POSTERIOR BACKOFF AND OPPORTUNITY SELECTION

The allocation menu is fixed by the calibration above, while the main training run uses online statistics to select opportunities and route individual prompts among these levels.

CALIPER maintains intervention outcomes over the most recent 32 optimizer steps. Statistics are organized hierarchically using the opportunity attributes $c = (d, z, \chi, f, \ell)$.

Starting from the complete five-component type, CALIPER uses the most specific statistics satisfying minimum observation counts of 24, 16, 8, and 4 for the five-, four-, three-, and two-component levels, respectively. If these thresholds are not met, it backs off to domain-level statistics. Each level uses a $\mathrm{Beta}(1, 1)$ prior.

Let μ_c, σ_c, and $\bar{T}_c$ denote the resulting posterior mean, standard deviation, and observed average suffix-generation length. The online opportunity-selection score is

$$S_{\text{online}}(a, z) = \frac{\min(1, \mu_c + 0.5\sigma_c)}{\bar{T}_c}. \tag{17}$$

When no suffix-length observation is available, CALIPER uses the feasible remaining generation budget as a conservative fallback estimate of generation cost. The highest-scoring opportunity determines the selected fork, with ties resolved in favor of the candidate with higher current-policy uncertainty.

A.4.5 Online Routing within the Calibrated Levels

For the selected opportunity, online routing uses

$$E_c^U = \frac{\min(1, \mu_c + 0.84\sigma_c)}{\bar{T}_c}, \qquad E_c^L = \frac{\max(0, \mu_c - 0.84\sigma_c)}{\bar{T}_c}. \tag{18}$$

The full-rollout reference efficiency is estimated from recent complete trajectories together with the four current initial roots. Its strict-success probability is Beta-smoothed, while its cost is the corresponding mean number of newly generated response tokens.

Cold Start. When the selected opportunity type has fewer than eight historical observations, CALIPER may allocate a $6/2$ probe through a globally shared cold-start allowance. At most two such prompt-level probes are admitted in one optimizer step; other prompts retain $8/0$ allocation.

Full-Exploration Regime. With sufficient history, if $E_c^U < E_{\text{full}}$, all remaining slots are assigned to complete trajectories, producing $8/0$.

Local-Favored Regime. After the 32-step warmup, CALIPER uses $4/4$ only when $n_c \geq 16$, $E_c^L > 1.10E_{\text{full}}$, and $\phi_W \geq 0.40$. Here, ϕ_W is the recent fraction of probed anchors for which the pivot and suffixes exhibit either strict terminal-outcome contrast or auxiliary failure-quality contrast.

Intermediate Regime. All remaining cases use $6/2$.

Current suffix outcomes are committed to the typed posterior only after allocation. Therefore, an outcome cannot be used to select the same suffix that produced it. Current initial-root outcomes may contribute to the full-rollout reference because these four mandatory trajectories have already been generated before routing.

A.4.6 Value- and Uncertainty-Oriented Interventions

At the selected fork, the value-oriented intervention is the feasible intervention with the highest score in Eq. equation 17.

CALIPER additionally selects an uncertainty-oriented intervention from the same restored execution state. Among alternatives to the value-oriented intervention, it favors higher posterior uncertainty and, as a secondary criterion, less frequently observed interventions.

For a $6/2$ allocation, the value-oriented and uncertainty-oriented interventions are sampled once each. For a $4/4$ allocation, each is sampled twice. If only one feasible intervention exists, the same intervention can fill both roles.

All local suffixes used in a same-state comparison branch from the same selected execution state.

A.4.7 Execution-State Restoration

CALIPER restores the actual execution state reached by the original trajectory rather than reconstructing an approximate textual prefix.

The stored state contains the preceding tokenized interaction context, previous tool observations, image inputs when present, consumed tool-call counts and signatures, duplicate-call state, assistant-turn and user/tool-response-turn counts, and the response mask at the fork.

Consequently, the pivot and its alternative suffixes share the same:

- task and tool definitions;
- pre-fork generated interaction history;
- tool observations;
- consumed interaction and tool budgets; and
- remaining generation budget.

Only the post-fork generation differs.

For conditional suffixes, copied prefix tokens have zero actor-loss mask. For the original pivot, global trajectory credit may still train the complete root trajectory, whereas intervention-derived local credit is restricted to the post-fork region.

A.4.8 Exact Local-Credit Calibration

Let m denote the number of suffixes generated at the selected fork and n_a^{hist} the backed-off historical count before the current local group. The probe-reliability factor is $r_a^{\text{probe}} = m/(m+2)$, while the historical-reliability factor is $r_a^{\text{hist}} = 0.5 + 0.5n_a^{\text{hist}}/(n_a^{\text{hist}} + 8)$.

The effective local-credit coefficient is

$$\lambda_a = \min\left(0.5,\ 0.5\frac{m}{m+2}\left(0.5 + 0.5\frac{n_a^{\text{hist}}}{n_a^{\text{hist}} + 8}\right)\right). \tag{19}$$

This corresponds to $\lambda_0 = 0.5$, $\lambda_{\max} = 0.5$, $\eta_p = 2$, $r_0 = 0.5$, and $\eta_h = 8$ in the notation of the main paper.

The pivot and its suffixes use the binary-outcome Leave-One-Out contrast defined in Eq. equation 7. Because local groups are small and rewards are binary, no local standard-deviation normalization is applied.

A.4.9 Failure-Quality Shaping

Strict terminal success remains the main task objective. Auxiliary quality is used only to differentiate unsuccessful trajectories. We set $q_i = 0$ for a strictly successful trajectory. For an unsuccessful trajectory, CALIPER uses

$$q_i = 0.50q_i^{\text{valid}} + 0.30q_i^{\text{prog}} + 0.10q_i^{\text{uncens}} + 0.10q_i^{\text{nondup}}. \tag{20}$$

Here, q_i^{valid} indicates whether the trajectory reaches a parseable final answer: a balanced final `\boxed{}` expression for mathematical or scientific tasks, or a complete Python code fence for code tasks.

The progress term q_i^{prog} is used only for code-generation tasks and is the bounded fraction of tests passed by an otherwise incorrect program. The non-censoring term q_i^{uncens} indicates that generation terminates without being stopped by a response, turn, or tool budget limit.

Let N_i^{tool} and N_i^{dup} denote the total and duplicate tool calls. The non-duplicate component is

$$q_i^{\text{nondup}} = 1 - \min\left(\frac{N_i^{\text{dup}}}{\max(N_i^{\text{tool}}, 1)}, 1\right). \tag{21}$$

Auxiliary values are Leave-One-Out centered only among failed samples in the corresponding comparison set. The resulting auxiliary contrast is multiplied by $\lambda_{\text{aux}} = 0.10$.

Auxiliary credit is additionally bounded by an absolute-mass cap. When the root strict-credit mass is nonzero, the auxiliary channel contributes at most 10% of the combined main-plus-auxiliary absolute advantage mass. When the root strict signal is zero, auxiliary credit is disabled rather than becoming the sole optimization objective.

A.4.10 COORDINATION OF GLOBAL AND LOCAL CREDIT

After constructing strict and auxiliary local credit, CALIPER limits the absolute contribution of the local channel whenever a nonzero root signal exists.

The local channel is capped at at most 12% of the combined root-plus-local absolute advantage mass. This is a ceiling rather than a target: local advantages are never artificially enlarged merely to reach the cap. A value of 0.08 is retained only as a diagnostic soft target and does not act as a lower bound or gating criterion.

When the root channel is zero, as in a homogeneous all-fail full-trajectory group, a relative root/local cap is undefined. CALIPER therefore retains verified strict local contrasts rather than suppressing them.

The final combined token-level advantage is clipped to $[-5, 5]$.

A.4.11 TARGETED NEGATIVE PPO UPDATES

The standard PPO clipping coefficients are $\epsilon_l = \epsilon_h = 0.2$. For locally inferior post-fork behavior, CALIPER uses $\epsilon_l^{\text{loc}} = 0.315$, while retaining $\epsilon_h^{\text{loc}} = 0.2$.

The implementation first marks post-fork tokens with negative local advantage. During PPO-loss computation, this local mask is further intersected with tokens whose final combined advantage is negative. Thus, the enlarged lower clipping interval is activated only when both $\bar{A}_{i,t}^{\text{loc}} < 0$ and $A_{i,t} < 0$.

For ordinary tokens, the minimum clipped importance ratio is 0.8; for targeted negative tokens, it becomes 0.685. The positive-update clipping bound remains 1.2. All prefix tokens and all tokens not identified as locally inferior retain the standard PPO clipping rule.

A.5 BASELINE IMPLEMENTATION AND FAIR COMPARISON

A.5.1 MATCHED ROLLOUT BUDGET

The central fairness constraint across all retrained methods is the total rollout-slot budget. Every method receives $B = 8$ rollout slots for each training prompt.

Any baseline-specific branch, suffix, intermediate-state resample, or auxiliary trajectory must consume one of these eight slots. Such samples are not generated in addition to the shared budget.

Thus, methods may differ in how they allocate or use rollout slots according to their original mechanisms, but not in the total number of slots available per prompt.

All methods are evaluated using the unified protocol in Appendix A.3.

A.5.2 QWEN3 AND PLAIN GRPO

The Qwen3-8B and Qwen3-4B rows evaluate their corresponding post-trained initialization checkpoints without task-specific reinforcement learning.

Plain GRPO uses all eight rollout slots for complete trajectories and applies group-relative optimization using strict terminal correctness. It does not use CALIPER's local opportunity construction, adaptive routing, same-state local credit, failure-quality local comparison, or targeted negative updates.

A.5.3 OTHER AGENTIC-RL BASELINES

We compare against DAPO, ARPO, AEPO, GiGPO, ELPO, and VinePPO. Their defining sampling or optimization mechanisms are retained, while any method-specific additional trajectories or intermediate-state samples are counted within the same $B = 8$ rollout-slot budget.

This prevents intermediate-state or adaptive-sampling methods from obtaining extra generations beyond those available to Plain GRPO or CALIPER.

A.5.4 ABLATION IMPLEMENTATIONS

The ablations in Table 2 remove one CALIPER component at a time while keeping the remaining training protocol unchanged.

w/o Adaptive Routing. We disable prompt-dependent routing and use a fixed $4/4$ root/suffix allocation throughout training. After the four mandatory initial complete trajectories, all four remaining rollout slots are therefore assigned to local suffixes rather than adaptively selecting among $8/0$, $6/2$, and $4/4$.

w/o Feedback-Driven Forks. We remove feedback-driven candidates $\mathcal{A}_i^{\mathrm{fb}}$ and construct local intervention opportunities only from autonomous decision forks $\mathcal{A}_i^{\mathrm{dec}}$.

w/o Matched Local Credit. We remove the matched same-state local-credit component while retaining the remaining CALIPER training pipeline and rollout budget.

w/o Opportunity Prioritization. We remove CALIPER's opportunity-prioritization component while retaining the same candidate opportunity space and rollout budget.

w/o Scope Alignment. We remove the restriction that intervention-derived local credit is confined to the post-fork behavior affected by the intervention, while retaining the remaining local-comparison mechanism.

w/o Targeted Negative Update. We disable the local-contrast-dependent lower clipping adjustment and use the standard PPO lower clipping coefficient for all tokens.

A.6 ADDITIONAL ANALYSIS PROTOCOLS

A.6.1 TERMINAL-OUTCOME HOMOGENEITY

Figure 1(a) analyzes Plain GRPO training groups containing eight complete trajectories for the same prompt.

A group is defined as *homogeneous* when all eight trajectories have the same strict terminal outcome. Both all-success and all-fail groups are counted as homogeneous, while the all-fail subset is additionally reported separately.

Because group-relative terminal credit becomes zero when all outcomes are identical, this statistic measures how frequently terminal supervision alone cannot distinguish trajectories within a training group.

A.6.2 TOOL-FEEDBACK CYCLES

For Figures 1(b) and 1(c), a completed tool-feedback cycle consists of a model tool call, execution by the environment, and the returned observation becoming available to a subsequent model decision.

For an all-fail evaluation group, Figure 1(b) records whether its failed trajectories complete different numbers of feedback cycles.

Figure 1(c) instead operates at the individual failed-trajectory level and records the number of completed feedback cycles before termination.

These statistics distinguish equality of terminal outcomes from equality of interaction processes.

A.6.3 TOOL-INTERACTION STRATIFICATION

For Figure 3(a), instance properties are defined using Qwen3-8B rollouts rather than CALIPER rollouts, avoiding method-dependent stratification.

For each evaluation instance, we compute the average number of tool calls and divide the instances into no-tool, light-tool-use (≤ 2), and heavy-tool-use (> 2) groups.

CALIPER and Plain GRPO are then compared on the same instances within each stratum. The reported differences are macro-averaged correctness differences.

Figure 3(b) additionally restricts the analysis to instances for which Qwen3-8B encounters tool-execution errors and categorizes their final outcomes using the recorded execution status.

A.6.4 Scope-Alignment Statistics

Figure 4 is computed over all 130 Qwen3-8B CALIPER training steps.

A training instance is counted as forming a valid local comparison when its selected matched pivot/suffix group yields nonzero strict or auxiliary local contrast. Prompt-level coverage is the fraction of training instances satisfying this condition.

For token-level coverage, the denominator consists of policy-generated response tokens. A token is counted as receiving local credit only when it lies in the post-fork local mask and receives nonzero intervention-derived advantage.

Copied suffix prefixes are excluded because their actor-loss masks are zero.

This distinguishes the fraction of prompts receiving additional local supervision from the fraction of generated behavior directly affected by that supervision.

A.6.5 Local Training-Signal Density

For Figure 5, we analyze failed decision forks actually selected for local probing during training.

Each selected fork is assigned to one of three mutually exclusive categories. A *strict rescue* occurs when the original pivot fails but at least one suffix succeeds. A *failure-quality contrast* occurs when the pivot and all suffixes fail under the strict verifier but differ in at least one auxiliary quality component, including final-answer validity, verifiable code progress, non-truncation, or non-duplicative tool behavior. All remaining probes are categorized as *no local contrast*.

The categorization is performed before auxiliary-credit magnitude gating. It therefore measures whether the probe exposes distinguishable local evidence rather than the final magnitude of its PPO update.

A.6.6 Compute-Efficiency Analysis

Figure 6 analyzes opportunity types observed at least eight times during the final 32 training steps.

Generation cost counts only newly generated suffix tokens. Copied prefix tokens are excluded because they are restored rather than regenerated.

For each opportunity type, strict-success efficiency is computed as the number of strict successes divided by the number of newly generated tokens. Full trajectories use the same definition. Each local opportunity type is normalized by the corresponding full-exploration efficiency, so $1\times$ indicates equal strict-success efficiency per newly generated token.

This analysis is descriptive rather than causal: it establishes that local opportunity types exhibit heterogeneous observed compute efficiency, which is consistent with the motivation for adaptive routing but does not by itself establish that routing causes the downstream performance gains.