CoRLA: From Next-State Feedback to Long-Horizon Credit for LLM Agent Training

Anonymous authors
Paper under double-blind review

Abstract

Next-state feedback provides dense supervision for large language model agents, but noisy local evaluations may not reflect an action's contribution to eventual task success. We propose **CoRLA** (Conformal Reward and Long-horizon Attribution), an uncertainty-aware reinforcement learning framework that connects reliable reward inference with long-horizon credit assignment. CoRLA uses conformal prediction to construct reward intervals from labels supplied by a frozen LLM teacher, updating calibration online as the policy evolves. The framework learns a potential from terminal returns under these interval constraints and propagates uncertainty-gated potential differences to assign long-horizon credit. Selective on-policy distillation adds complementary token-level supervision through hindsight corrections from low-uncertainty transitions. Policy optimization integrates local progress, long-horizon credit, and token-level supervision. Experiments show that CoRLA exceeds OpenClaw-RL by 11.2 percentage points in Terminal Bench 2.0 success rate, 8.2 points in SWE-bench Verified resolved rate, and 10.0 points in AIME 24/25 accuracy. Compared with local process rewards, CoRLA's credits also align more closely with return changes under sampled action replacements. The code is available at `https://anonymous.4open.science/r/CoRLA-BF10`.

1 Introduction

Large language model (LLM) agents solve tasks by interleaving reasoning with actions and incorporating observations from their environment (Yao et al., 2022). This interaction paradigm supports software repair, terminal operation, and tool-assisted problem solving, where success depends on a sequence of decisions rather than a single response. Reinforcement learning (RL) offers a way to improve these decisions through interaction, but the available feedback varies in both timing and reliability. A final task outcome provides supervision for the entire trajectory, whereas a tool response, execution error, or test result provides immediate evidence about the preceding action. Such *next-state signals* can support learning throughout an interaction (Wang et al., 2026). However, immediate feedback does not necessarily reveal an action's contribution to eventual success: a failed command may provide useful diagnostic information, while a successful execution may still advance an incorrect plan. Learning from these signals therefore requires resolving two coupled questions: how much to trust the inferred reward, and how to assign credit over the task horizon.

Relevant approaches can be organized into three complementary lines. *Outcome-based RL*, exemplified by ReTool (Feng et al., 2026a), learns multi-turn tool-use strategies from final task feedback. This avoids intermediate annotation, but a terminal score alone does not distinguish decisive actions from incidental ones. *Process supervision and feedback-based learning* provide denser guidance: process reward models assess intermediate reasoning steps (Lightman et al., 2024), while OpenClaw-RL extracts scalar evaluations and hindsight-guided on-policy distillation (OPD) signals from next-state observations (Wang et al., 2026). Nevertheless, step judgments can be ambiguous, as illustrated by the difficulty of distinguishing neutral from erroneous tool actions in AgentProcessBench (Fan et al., 2026); a local assessment also need not capture delayed utility. *Long-horizon credit assignment* addresses temporal attribution through return decomposition in RUDDER (Arjona-Medina et al., 2019), retrospective action assessment in HCAPO (Tan et al., 2026), and topology-aware reward propagation in RewardFlow (Feng et al., 2026b). These methods motivate richer attribution, but leave a distinct question for learning from noisy next-state evaluations: *how should predictive uncertainty in a process reward govern its influence on temporally extended policy updates?*

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

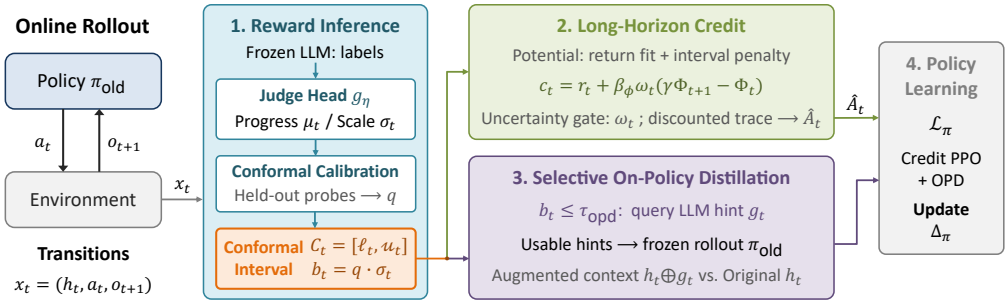

Figure 1: The architecture of our proposed CoRLA for training LLM agents.

To address this challenge, we propose **CoRLA** (*Conformal Reward and Long-horizon Attribution*), an uncertainty-aware framework for learning from noisy next-state signals. As illustrated in Figure 1, CoRLA constructs reward intervals by calibrating predictions against labels from a frozen LLM teacher (Lei et al., 2018). Calibration is updated online as the policy evolves; nominal coverage is not guaranteed in this adaptive setting. Since local progress can diverge from eventual success, CoRLA learns a potential from terminal returns under reward-interval constraints. An uncertainty gate downweights potential differences for transitions with wider intervals before propagating credit across actions. Selective OPD complements this temporal supervision with actionable hindsight corrections from low-uncertainty transitions, guiding token choices on the policy's own trajectories. Policy optimization combines local progress, long-horizon credit, and token-level supervision.

We evaluate terminal operation on Terminal Bench 2.0 (Merrill et al., 2026), software repair on SWE-bench Verified (OpenAI, 2024), and tool-assisted mathematical reasoning on AIME 2024 and 2025 (MAA, 2024; 2025). Relative to OpenClaw-RL, the reported scores improve by 11.2 percentage points on Terminal-Bench, 8.2 on SWE-bench, and 10.0 on each AIME evaluation. Further analyses suggest that long-horizon credit provides larger gains on longer tasks. Counterfactual tests also show that CoRLA identifies critical actions more accurately than local process rewards.

2 Approach

For a task drawn from $\mathcal{D}$ with initial observation o_1, an LLM-based agent generates a trajectory $\tau = (o_1, a_1, \ldots, a_T, o_{T+1})$. At step t, it samples $a_t \sim \pi(\cdot \mid h_t)$ from the interaction history where $h_t = (o_1, a_1, \ldots, a_{t-1}, o_t)$ and receives feedback o_{t+1}. Our approach integrates reliable reward inference, uncertainty-aware credit assignment, selective OPD, and policy learning (Sections 2.1–2.4).

2.1 Reliable Reward Inference with Conformal Prediction

Next-state feedback provides evidence about local progress, but teacher-derived process labels can be biased and have transition-dependent errors (Wang et al., 2026). Point estimates alone provide limited information about the reliability of this supervision. Therefore, we use conformal prediction to calibrate process-label intervals from observed prediction errors (Lei et al., 2018).

Predicting local progress and uncertainty. We use a pretrained LLM F_{θ_0} with a frozen backbone and language-model head, three separate LoRA adapters (Hu et al., 2021), and two lightweight MLP heads. The policy uses Δ_π; the judge and potential use (Δ_J, g_η) and (Δ_Φ, g_ψ), respectively. Each forward pass activates one adapter, and the adapters share no trainable parameters. The judge reads the representation after o_{t+1} to predict progress μ_t and scale σ_t for transition $x_t = (h_t, a_t, o_{t+1})$:

$$\begin{aligned} z_t^{\text{trans}} &= F_{\theta_0+\Delta_c}(x_t), \qquad (m_t, v_t) = g_\eta(z_t^{\text{trans}}), \\ \mu_t &= \tanh(m_t), \qquad \sigma_t = \sqrt{\sigma_{\min}^2 + \text{softplus}(v_t)}, \end{aligned} \tag{1}$$

where $\sigma_{\min} > 0$. The location $\mu_t \in [-1, 1]$ predicts a process label, and σ_t expresses relative predictive uncertainty. Calibration below converts this scale into an interval radius.

Bootstrapping the judge from teacher labels. The judge initially lacks progress supervision, so we first train it with a warm start to provide estimates for subsequent online RL. A frozen LLM

teacher labels initial-policy transitions as $\tilde{r}_t = J_{\mathrm{LLM}}(x_t) \in [-1, 1]$ under a fixed prompt and decoding protocol (Wang et al., 2026). These labels $\tilde{r}_t$ serve as noisy but informative pseudo progress assessments. We optimize (Δ_c, η) on a collected transition dataset $\mathcal{D}_J^{(0)} = \{(x_t, \tilde{r}_t)\}$ with

$$\mathcal{L}_{\mathrm{NS}}(\mathcal{D}_J) = \frac{1}{|\mathcal{D}_J|} \sum_{(x,\tilde{r})\in\mathcal{D}_J} \left[\frac{\rho_\delta(\mu(x) - \tilde{r})}{2\sigma(x)^2} + \frac{1}{2}\log\sigma(x)^2 \right], \tag{2}$$

where $\rho_\delta(e) = e^2/2$ for $|e| \le \delta$ and $\delta(|e| - \delta/2)$ otherwise. This location-scale objective reduces the influence of large residuals, while the log-scale term penalizes uniformly inflated uncertainty.

Refreshing the judge with selective teacher queries. As the policy changes during RL, it visits new transitions on which the bootstrapped judge may be inaccurate. Instead of using all transitions, we refresh the judge through uncertainty-guided selective teacher querying. At the k-th iteration, trajectories yield N new transitions $\mathcal{D}_{\mathrm{new}}$. We first uniformly draw M_{cal} calibration probes from these transitions for reservation and then select the K most uncertain transitions by their predicted scales σ for querying their labels from the frozen teacher LLM:

$$\mathcal{S}^{\mathrm{cal}} \sim \mathrm{Uniform}(\mathcal{D}_{\mathrm{new}}), \quad \mathcal{S}^{\mathrm{judge}} = \underset{x_i\in\mathcal{D}_{\mathrm{new}}\setminus\mathcal{S}^{\mathrm{cal}}}{\text{Top-K}} \sigma(x_i), \quad \mathcal{D}_J^{(k)} = \{(x_i, J_{\mathrm{LLM}}(x_i)) : x_i \in \mathcal{S}^{\mathrm{judge}}\}.$$

The two sets $\mathcal{S}^{\mathrm{cal}}, \mathcal{S}^{\mathrm{judge}}$ are disjoint by construction. To retain earlier supervision, we add $\mathcal{D}_J^{(k)}$ to a judge replay buffer $\mathcal{B}^{\mathrm{judge}}$ initialized with $\mathcal{D}_J^{(0)}$. Mini-batches mix newly added pairs with uniformly sampled historical pairs; calibration probes never enter this buffer. We update (Δ_J, η) by minimizing $\mathcal{L}_{\mathrm{NS}}$ in Eq. (2).

Constructing teacher-based online conformal intervals. The judge's predicted scale is not calibrated to a coverage level, so we use conformal prediction to determine interval widths from observed prediction errors. We query the frozen teacher for labels $\tilde{r}_i = J_{\mathrm{LLM}}(x_i)$ on the reserved probes $x_i \in \mathcal{S}^{\mathrm{cal}}$. These labels are used for calibration and excluded from judge regression and replay. We retain recent probe inputs and labels in $\mathcal{B}^{\mathrm{teach}}$, initialized with the held-out bootstrap subset. After each round of judge updates, we rescore this buffer with the current predictor (μ, σ). For $n = |\mathcal{B}^{\mathrm{teach}}|$, the normalized nonconformity scores and conformal quantile are

$$e_i^{\mathrm{teach}} = \frac{|\tilde{r}_i - \mu(x_i)|}{\sigma(x_i)}, \qquad q^{\mathrm{teach}} = e^{\mathrm{teach}}_{(\lceil (n+1)(1-\alpha_{\mathrm{teach}})\rceil)} \tag{3}$$

where $\alpha_{\mathrm{teach}} \in (0, 1)$, $e^{\mathrm{teach}}_{(j)}$ is the j-th ordered score, and $e^{\mathrm{teach}}_{(n+1)} = +\infty$. We construct

$$b_t = q^{\mathrm{teach}}\sigma(x_t), \qquad C_t^{\mathrm{teach}} = [\ell_t, u_t] = [\mu(x_t) - b_t, \mu(x_t) + b_t]. \tag{4}$$

We cache these intervals for potential learning (Section 2.2), with wider intervals imposing weaker local constraints. The judge adapter and head remain fixed during potential learning, preventing these updates from directly changing the judge's predictions. Adaptive RL need not satisfy standard split-conformal assumptions, so we do not claim nominal coverage guarantees (Barber et al., 2023).

2.2 Uncertainty-aware Long-Horizon Credit Assignment

Learning a potential from terminal outcomes. Local progress does not directly measure an action's eventual utility. For a completed trajectory with terminal outcome $R_T \in [0, 1]$, we define the environment reward and discounted return as

$$r_t^{\mathrm{env}} = \mathbf{1}\{t = T\}R_T, \qquad G_t = \sum_{s=t}^{T} \gamma^{s-t} r_s^{\mathrm{env}} = \gamma^{T-t} R_T, \tag{5}$$

where $\gamma \in (0, 1]$ is the discount factor. We learn a history-dependent potential Φ_t from G_t using g_ψ over the potential adapter Δ_Φ. Its representation uses only the history h_t available before action a_t:

$$z_t^{\mathrm{state}} = F_{\theta_0+\Delta_c}(h_t), \qquad \Phi_t = g_\psi(z_t^{\mathrm{state}}), \qquad \mathcal{L}_{\mathrm{return}} = \frac{1}{N}\sum_{\tau\in\mathcal{D}_\tau}\sum_{t=1}^{T} \rho_\delta(\Phi_t - G_t). \tag{6}$$

Across trajectories, this supervision relates each prefix to discounted future success. Specially, we set $\Phi(h_{T+1}) = 0$ at a true terminal state by masking the output, since no future reward remains after the terminal outcome has been issued.

Regularizing potential increments with teacher intervals. Terminal returns anchor the potential, while teacher intervals provide auxiliary guidance on local progress. For the discounted increment $\Delta\Phi_t = \gamma\Phi_{t+1} - \Phi_t$, let $\widehat{\ell}_t = \mathrm{sg}(\ell_t)$ and $\widehat{u}_t = \mathrm{sg}(u_t)$ be the cached endpoints from Section 2.1, where sg denotes stop-gradient. We use the soft interval penalty

$$\mathcal{L}_{\text{CP-prog}} = \frac{1}{N}\sum_{\tau\in\mathcal{D}_\tau}\sum_{t=1}^{T-1}\left(\max\{0, \widehat{\ell}_t - \Delta\Phi_t\}^2 + \max\{0, \Delta\Phi_t - \widehat{u}_t\}^2\right), \tag{7}$$
$$\mathcal{L}_\Phi = \mathcal{L}_{\text{return}} + \lambda_{\text{CP}}\mathcal{L}_{\text{CP-prog}}, \qquad \lambda_{\text{CP}} \geq 0.$$

We optimize only (Δ_Φ, ψ), keeping the judge, policy, shared backbone, and cached endpoints fixed. The penalty acts only outside the interval; at a fixed center, wider intervals impose weaker constraints. Applying these endpoints directly assumes that the teacher's scoring rubric is compatible with discounted return increments; matching numerical ranges alone is insufficient. In particular, exact fitting $\Phi_t = G_t$ along a completed trajectory gives $\Delta\Phi_t = 0$ for $t < T$, which can conflict with a positive local-progress label. Thus, λ_{CP} controls a tradeoff between two supervision signals, rather than resolving their semantic or scale mismatch. Calibration of teacher labels does not establish coverage for $\Delta\Phi_t$. We exclude $t = T$, where $\Delta\Phi_T = -\Phi_T$ can conflict with a positive completion label; the terminal outcome still enters the return loss and policy credit.

Uncertainty-aware credit assignment. To control how strongly uncertain transitions influence policy learning, we normalize the conformal interval width by the range of process labels. Let $\mathcal{R}$ denote the radius of this range, with $\mathcal{R} = 1$ for labels in $[-1, 1]$. Using $u_t - \ell_t = 2b_t$, we define

$$\omega_t = \mathrm{sg}\left[1 - \mathrm{clip}\left(\frac{u_t - \ell_t}{2\mathcal{R}}, 0, 1\right)\right] = \mathrm{sg}\left[1 - \mathrm{clip}\left(\frac{b_t}{\mathcal{R}}, 0, 1\right)\right]. \tag{8}$$

The gate lies in $[0, 1]$, decreases linearly with interval half-width, and reaches zero when $b_t \geq \mathcal{R}$. It uses teacher-label uncertainty as a reliability proxy for the correction, without estimating potential uncertainty. Its independence from the judge mean allows a precise near-zero progress estimate to retain a large weight. We use the untruncated interval width and hold the gate fixed throughout the round. After updating (Δ_Φ, ψ), we recompute Φ_t and form the step credit:

$$c_t = \mathrm{sg}\left[r_t^{\text{env}} + \beta_\Phi\omega_t\left(\gamma\Phi_{t+1} - \Phi_t\right)\right], \qquad \beta_\Phi > 0. \tag{9}$$

The environment reward r_t^{env} (denoted in Eq. (5)) retains unit weight; the gate scales the potential correction. We cache the credits for policy learning, with $\Phi_{T+1} = 0$ at true termination and a retained bootstrap potential at nonterminal truncations. Later credits propagate to earlier actions through a GAE-style discounted trace (Schulman et al., 2015):

$$\widehat{A}_t = \mathrm{sg}\left[\sum_{s=0}^{T-t}(\gamma\lambda_A)^s c_{t+s}\right], \qquad 0 \leq \lambda_A \leq 1. \tag{10}$$

Here λ_A controls the contribution of later credits. When $\beta_\Phi\omega_t = 1$ at every step, c_t is the temporal-difference residual of Φ, and the trace has the standard GAE form. With transition-dependent gates, potential differences generally no longer telescope, so the shaping term lacks the usual policy-invariance guarantee (Ng et al., 1999). We therefore use $\widehat{A}_t$ as a surrogate credit signal, without claiming an unbiased task-advantage estimate or a convergence guarantee.

2.3 Selective On-Policy Distillation

Selecting transitions and extracting hindsight hints. We use corrective next-state feedback to guide the policy's token choices through hindsight-conditioned OPD (Wang et al., 2026). We first select transitions whose cached conformal half-width satisfies $\widehat{b}_t \leq \tau_{\text{OPD}}$, where $\widehat{b}_t = \mathrm{sg}(b_t)$ and $\tau_{\text{OPD}} \geq 0$ is a finite threshold. This applies a common width criterion across rounds; the selected fraction can vary as the conformal intervals change. For each selected transition, the frozen LLM teacher extracts a concise correction g_t from (h_t, a_t, o_{t+1}) using a fixed hint-extraction prompt. We set $g_t = \varnothing$ when no actionable correction is found or the transition is not selected. The augmented teacher context and OPD mask are

$$h_t^+ = h_t \oplus g_t, \qquad m_t^{\text{OPD}} = \mathbf{1}\{\widehat{b}_t \leq \tau_{\text{OPD}}\}\mathbf{1}\{g_t \neq \varnothing\}. \tag{11}$$

The augmented context supervises the already sampled action; the student uses only h_t when choosing a_t. Interval width serves as a reliability proxy and does not guarantee hint correctness. Unselected transitions still contribute to potential learning and outcome-based policy updates.

Constructing token-level distillation targets. We freeze a copy π_{old} of the rollout policy for the current update. For turns with $m_t^{\text{OPD}} = 1$, this model scores the sampled action under both h_t^+ and h_t, providing the hint-conditioned teacher and reference probabilities, respectively. Both probabilities condition on the same sampled token prefix $a_{t,<j}$. For token $a_{t,j}$ in an L_t-token action a_t, we cache

$$\widehat{A}_{t,j}^{\text{OPD}} = \text{sg}\left[\text{clip}\left(\log\left(\frac{\pi_{\text{old}}(a_{t,j} \mid h_t^+, a_{t,<j})}{\pi_{\text{old}}(a_{t,j} \mid h_t, a_{t,<j})}\right), -C_{\text{OPD}}, C_{\text{OPD}}\right)\right], \tag{12}$$

where $C_{\text{OPD}} > 0$ bounds the log-probability gap. A positive target favors a sampled token that becomes more probable with the correction; a negative target discourages it. Unselected turns have zero OPD target and require no hint-conditioned scoring pass. This sampled-token formulation distills on the policy's own trajectories and adds no trainable teacher or auxiliary head. The masks and token targets remain fixed during the policy update.

2.4 Final Policy Learning

Combining temporal credit and selective distillation. For policy optimization, we assign each policy-generated token in a_t the fixed credit-based advantage $\widehat{A}_t$ from Eq. (10), without subtracting another potential baseline. Let $\chi_{t,j} = (h_t, a_{t,<j})$ and define the token probability ratio $\varrho_{t,j} = \pi_{\Delta_\pi}(a_{t,j} \mid \chi_{t,j}) / \pi_{\text{old}}(a_{t,j} \mid \chi_{t,j})$. Using the PPO clipped surrogate (Schulman et al., 2017),

$$\mathcal{S}_{\epsilon_-,\epsilon_+}(\varrho, A) = \min\{\varrho A, \text{clip}(\varrho, 1-\epsilon_-, 1+\epsilon_+)A\}, \tag{13}$$

where $\epsilon_-, \epsilon_+ > 0$ control the lower and upper clipping bounds, respectively. The combined loss is

$$\mathcal{L}_\pi = -\frac{1}{|\mathcal{D}_\tau|} \sum_{\tau \in \mathcal{D}_\tau} \sum_{t=1}^{T} \gamma^{t-1} \sum_{j=1}^{L_t} \Big[\mathcal{S}_{\epsilon_-,\epsilon_+}(\varrho_{t,j}, \widehat{A}_t) + \lambda_{\text{OPD}} m_t^{\text{OPD}} \mathcal{S}_{\epsilon_-,\epsilon_+}(\varrho_{t,j}, \widehat{A}_{t,j}^{\text{OPD}})\Big], \tag{14}$$

where $\lambda_{\text{OPD}} \geq 0$ controls distillation. We average over trajectories and sum over agent-generated action tokens without length normalization, clipping the two terms separately. Gradients update only Δ_π; the backbone, language-model head, judge and potential branches, masks, and cached targets remain fixed. We additionally apply KL regularization toward the frozen initial policy.

3 Experiments

3.1 Experimental Setup

Datasets and Metrics. We conduct experiments on four datasets across three task domains. *Terminal Bench 2.0* (Merrill et al., 2026) contains 89 tasks in executable terminal environments, with task-specific tests verifying completion. *SWE-bench Verified* (OpenAI, 2024) consists of 500 human-validated software issues, where an agent must modify a repository to satisfy the issue's evaluation tests. *AIME 2024* (MAA, 2024) and *AIME 2025* (MAA, 2025) evaluate mathematical problem solving in a tool-assisted setting, containing 30 and 30 evaluation problems, respectively. We report *Task Success Rate* on Terminal Bench 2.0, *Resolved Rate* on SWE-bench Verified, and *Accuracy* on AIME, all as percentages. For reward-noise analysis, we use *AgentProcessBench* (Fan et al., 2026), which provides human annotations of step-level effectiveness in tool-using trajectories. Its three-way labels, $y \in \{-1, 0, +1\}$, serve as the clean reference annotations in the controlled corruption experiments.

Baselines. We compare our method against a suite of baselines covering closed-source API models and open-source fine-tuned models. Our main algorithmic direct comparator is **OpenClaw-RL** (Wang et al., 2026), which learns from evaluative and directive signals extracted from next-state feedback. To contextualize absolute task performance, we also compare against **Claude Sonnet 4.5** (Anthropic, 2025b), **Claude Opus 4.1** (Anthropic, 2025a), **GPT-5.2** (OpenAI, 2025b), **GPT-5** (OpenAI, 2025a), **GPT-5-Mini** (OpenAI, 2025a), **GPT-OSS-120B** (OpenAI, 2025c), **Kimi K2 Instruct** (Moonshot AI, 2025), **Qwen3 Coder 480B** (Qwen Team, 2025), and **Qwen3 Coder 30B A3B** (Qwen Team, 2025). Implementation details can refer to Appendix B.

Table 1: Main task results (%; higher is better). #Tokens (K) denotes the average number of input and output tokens per task. Bold marks our method outperforms OpenClaw-RL.

Model	Terminal Bench 2.0		SWE-bench Verified		AIME 2024	AIME 2025
	Success Rate	#Tokens	Resolve Rate	#Tokens	Accuracy	Accuracy
Claude Sonnet 4.5	40.4	11.2	71.4	21.1	93.3	86.7
Claude Opus 4.1	34.8	13.7	67.6	20.6	83.3	80.0
GPT-5.2	62.9	25.2	72.8	34.4	96.7	96.7
GPT-5	49.4	23.3	65.0	32.1	93.3	90.0
GPT-5-Mini	31.5	20.6	56.2	23.5	86.7	83.3
GPT-OSS-120B	19.1	14.1	26.0	24.1	96.7	93.3
Kimi K2 Instruct	28.1	26.4	53.4	38.9	70.0	56.7
Qwen3 Coder 480B	28.1	22.2	55.4	33.3	56.7	40.0
Qwen3 Coder 30B A3B	22.5	18.5	51.6	21.8	73.3	56.7
OpenClaw-RL	13.5	11.5	15.4	21.1	26.7	23.3
CoRLA (ours)	**24.7**	14.7	**23.6**	20.7	**36.7**	**33.3**

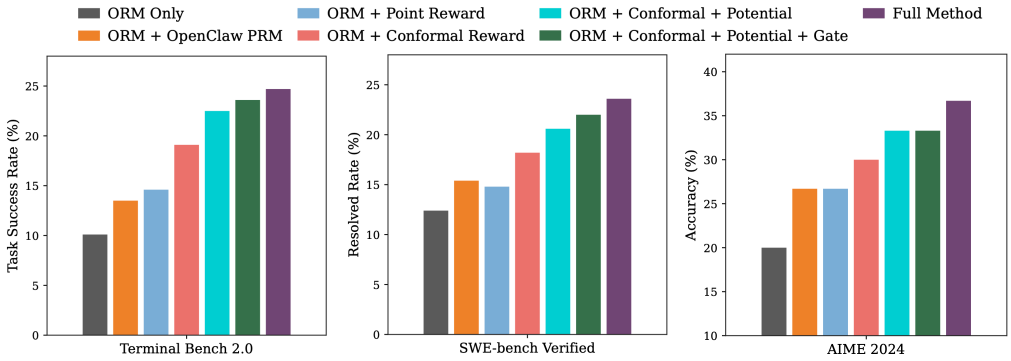

Figure 2: Ablation results for our method and six variants across the three task domains.

3.2 Main Experiments

Our method outperforms OpenClaw-RL on all four evaluations in Table 1. Terminal Bench 2.0 success rate increases from 13.5% to 24.7%, yielding the largest absolute gain of 11.2 pp. SWE-bench Verified resolved rate improves from 15.4% to 23.6%, a gain of 8.2 pp. AIME accuracy reaches 36.7% in 2024 and 33.3% in 2025, exceeding OpenClaw-RL by 10.0 pp in both years. These gains extend across terminal interaction, repository repair, and tool-assisted mathematical reasoning, supporting the effectiveness of the full framework across three tasks.

On Terminal Bench, higher success comes with an increase in average token usage from 11.5K to 14.7K per task, approximately 27.8% more than OpenClaw-RL. On SWE-bench, average usage decreases from 21.1K to 20.7K tokens, approximately 1.9% fewer, alongside the higher resolved rate. Thus, CoRLA achieves higher task performance on both benchmarks, while changes in token usage vary across tasks.

On Terminal Bench, our method exceeds GPT-OSS-120B and Qwen3 Coder 30B A3B by 5.6 and 2.2 pp, respectively. SWE-bench and AIME performance remains below every listed reference model, indicating room to improve absolute task capability. GPT-5.2 also achieves higher performance with fewer tokens on both agent benchmarks. These reference results contextualize the gains over OpenClaw-RL, although differences in model families and scales prevent attributing cross-model performance gaps solely to the training algorithm

3.3 Ablation Analysis

We compare seven configurations to examine the contributions of reward inference, temporal credit, and selective distillation (Fig. 2). *ORM Only* trains with only outcome rewards. *ORM + OpenClaw PRM* adds process rewards inferred directly from next-state feedback as in (Wang et al., 2026). *ORM*

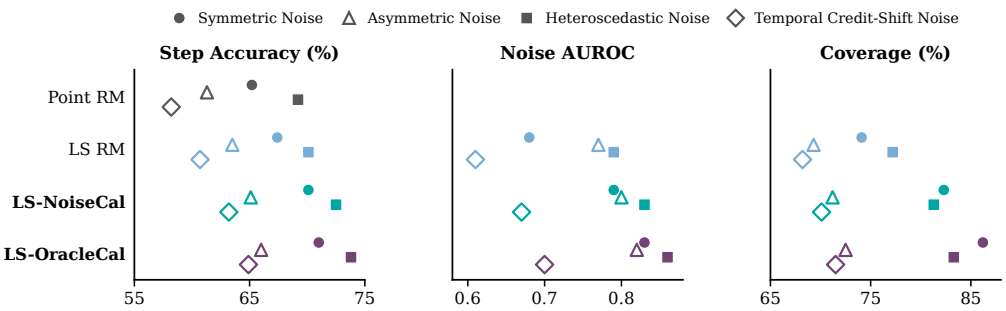

Figure 3: Reward inference results across four variants under four forms of label corruption.

+ *Point Reward* instead uses point estimates from our reward predictor, while *ORM + Conformal Reward* introduces conformal reward intervals. Starting from the interval-reward variant, *ORM + Conformal + Potential* adds long-horizon credit derived from the learned potential, and *ORM + Conformal + Potential + Gate* further weights the potential differences by their uncertainty gates. Finally, *Full Method* adds selective OPD to provide token-level guidance on selected transitions.

The progressive ablations in Fig. 2 associate the gains with both reward inference and temporal credit. Relative to Point Reward, the interval-reward variant improves Terminal, SWE, and Tool-Call performance by 4.5, 3.4, and 3.3 pp. Adding the potential increases these scores by another 3.4, 2.4, and 3.3 pp. The uncertainty gate then adds 1.1 pp on Terminal and 1.4 pp on SWE, while Tool-Call remains at 33.3%. Finally, selective OPD increases the three scores by 1.1, 1.6, and 3.4 pp. The full method therefore improves on ORM Only by 14.6, 11.2, and 16.7 pp. These are incremental effects along the tested sequence of variants; they do not isolate all possible interactions among components.

3.4 Reward Noise Robustness Analysis

We split AgentProcessBench into training, calibration, and test subsets (8:1:1), retaining clean test labels while applying four types of training-label corruption. *Symmetric noise* flips a label with probability ρ, choosing uniformly between the other two classes. *Asymmetric noise* permits only adjacent-class flips with probability ρ, choosing uniformly between -1 and $+1$ when the original label is 0. *Heteroscedastic noise* applies symmetric corruption only to ambiguous steps, identified by disagreement across repeated evaluations of the same judge at different temperatures or seeds. *Temporal credit-shift noise* delays the first error label to a later step.

We compare Point RM, an uncalibrated location-scale (LS) RM, and two conformalized LS variants. Point RM uses only the predicted reward location, while LS RM additionally estimates a sample-dependent scale but does not apply conformal calibration. LS-NoiseCal constructs LS conformal intervals using noisy calibration labels, whereas LS-OracleCal uses clean calibration labels as a diagnostic oracle, separate from the teacher-based online calibration. Coverage measures the fraction of clean test labels contained in the calibrated reward intervals, while Noise AUROC evaluates how well the predicted uncertainty score ranks corrupted steps above uncorrupted ones.

LS-NoiseCal exceeds LS RM on all three metrics under every noise mechanism (Fig. 3). Under symmetric noise, StepAcc increases from 67.4% to 70.1%, Noise AUROC from 0.68 to 0.79, and coverage from 74.1% to 82.3%. LS-OracleCal achieves the highest scores in every setting. These comparisons concern estimator variants; attributing StepAcc gains to calibration requires specifying the label-decision rule. Temporal-shift noise remains the most challenging, with LS-OracleCal reaching only 64.9% StepAcc and 0.70 AUROC. This residual difficulty motivates the temporal-credit evaluation in Section 3.5.

3.5 Long-Horizon Credit Assignment Analysis

Performance gain versus trajectory length. We compare the full model with a variant without long-horizon credit, grouping tasks by base-model trajectory length at the 33rd and 67th percentiles. This fixes the short, medium, and long groups independently of the evaluated policies. The full model shows larger gains on long than short tasks in all three benchmarks (Fig. 4). To quantify this trend across individual tasks, we further compute $\rho = \mathrm{SpearmanCorr}_i(T_i, \Delta_i)$ for each benchmark. Here,

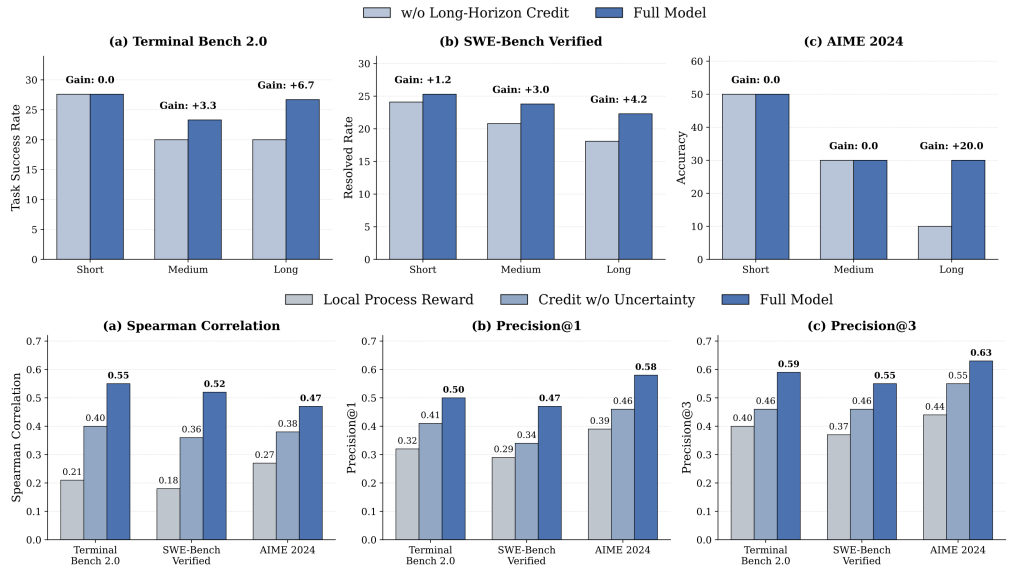

Figure 4: Performance by trajectory length (Upper). Counterfactual credit attribution results (Lower).

T_i is the base-model trajectory length, and Δ_i is the full model performance gain over the variant on the i-th task. The correlations are 0.38, 0.42, and 0.25 for Terminal Bench 2.0, SWE-bench Verified, and AIME 2024, respectively. All three are positive, supporting the trend that using long-horizon credit tends to yield larger gains on longer trajectories.

Counterfactual credit attribution. To test whether a high credit identifies a critical action, we examine successful trajectories from Terminal Bench and AIME 2024 and a 100-task subset of SWE-bench Verified. Steps are sampled from early, middle, and late portions of a trajectory. At a sampled step t, we keep the preceding history fixed, replace the original action with M counterfactual actions, and rollout each continuation to termination. Let $G(\tau)$ denote the trajectory's task return, and $\tau_t^{(j)}$ the j-th continuation. The estimated contribution of the original action is $\Delta G_t = G(\tau) - \frac{1}{M}\sum_{j=1}^{M} G(\tau_t^{(j)})$. A larger ΔG_t means that replacing the action t caused a larger return decrease. Spearman correlation measures rank agreement between credit c_t and ΔG_t. For critical-step coverage evaluation, let $\mathcal{C}_K^\star = \text{Top-K}(\Delta G_t)$ and $\widehat{\mathcal{C}}_K = \text{Top-K}(c_t)$ within a trajectory. We report $\text{Precision@}K = \frac{|\widehat{\mathcal{C}}_K \cap \mathcal{C}_K^\star|}{K}, K \in \{1, 3\}$. In Fig. 4, the full model achieves the highest correlation and critical-step precision on all three benchmarks. Its Spearman correlations are 0.55, 0.52, and 0.47 on terminal, SWE, and AIME, compared with 0.21, 0.18, and 0.27 for local process reward. It also exceeds credit w/o uncertainty by 0.15, 0.16, and 0.09 in correlation. Relative to that comparator, Precision@1 increases by 0.09, 0.13, and 0.12, and Precision@3 by 0.13, 0.09, and 0.08. Thus, the uncertainty-aware credits align more closely with the measured effects of replacing actions, complementing the task-level gains on longer trajectories.

Table 2: Task performance for different gating mechanism.

Benchmark	No Gate	Raw-σ Gate	Hard Filtering Gate	CP Soft Gate
Terminal Bench 2.0	22.5	23.6	21.3	**24.7**
SWE-bench Verified	21.0	22.2	21.4	**23.6**
AIME 2024	33.3	36.7	30.0	**36.7**

3.6 Uncertainty-Aware Gating Analysis

Gate variants. We compare four variants to test the roles of gating, conformal scaling, and soft weighting. All variants retain the same judge training, conformal potential constraints, and selective

OPD, changing only the weight on the potential difference in Eq. (9):

$$\omega_t^{(v)} = \begin{cases} 1, & v = \text{No Gate}, \\ 1 - \text{clip}(\sigma(x_t)/R, 0, 1), & v = \text{Raw-}\sigma\text{ Gate}, \\ \mathbf{1}\{b_t/\mathcal{R} \le \tau_g\}, & v = \text{Hard Filtering Gate}, \\ 1 - \text{clip}(b_t/\mathcal{R}, 0, 1), & v = \text{CP Soft Gate}, \end{cases} \tag{15}$$

where $\mathcal{R} = 1$ and $\tau_g \in [0, 1]$ is a threshold. CP Soft Gate is the proposed gate in Eq. (8). Raw-σ Gate omits conformal scaling only in the gate; hard filtering suppresses the potential-difference term on uncertain transitions. Environment rewards and OPD contributions remain active, and all gate values are detached during policy updates.

CP Soft Gate achieves the best or tied-best performance across all three benchmarks (Table 2). It surpasses Raw-σ Gate by 1.1 pp on Terminal Bench and 1.4 pp on SWE-bench, while tying on AIME 2024. It also exceeds No Gate and Hard Filtering Gate by 2.2–3.4 pp and 2.2–6.7 pp, respectively. These comparisons support using conformally calibrated soft weights to regulate how potential differences contribute to policy learning.

4 Related Work

Agentic RL and Tool Use. LLM agents couple language generation with environment interaction, as exemplified by interleaving of reasoning and acting in ReAct (Yao et al., 2022) and self-supervised API use in Toolformer (Schick et al., 2023). Recent RL methods optimize these interaction strategies from task feedback: Search-R1 (Jin et al., 2025) trains multi-turn search with outcome rewards, ReTool (Feng et al., 2026a) develops code-integrated reasoning through RL, and ToolRL (Qian et al., 2026) studies reward design for tool selection and execution. As trajectories lengthen, terminal rewards provide weak temporal attribution; HCAPO (Tan et al., 2026) addresses this with hindsight-refined action values, while RewardFlow (Feng et al., 2026b) propagates reward over state graphs. OpenClaw-RL (Wang et al., 2026) instead derives both scalar evaluations and directive OPD signals from next-state observations. CoRLA builds on this feedback-rich setting but targets the reliability of the derived supervision: it converts teacher-labeled next-state feedback into conformal reward intervals, uses terminal returns and interval constraints to learn long-horizon credit, and restricts token-level hindsight guidance to selected transitions.

Process Reward Modeling. Process reward models provide step-level supervision beyond terminal outcome labels. Previous work (Lightman et al., 2024) trains process verifiers from human annotations, Math-Shepherd (Wang et al., 2024) obtains step labels from continuation outcomes, and Rewarding Progress (Setlur et al., 2025) measures a step by its change in the probability of future success. Extending process supervision to agents is harder because intermediate tool actions need not have an immediately verifiable notion of correctness; AgentProcessBench (Fan et al., 2026) documents this ambiguity, AgentPRM (Xi et al., 2026) models both future promise and local progress, and online PRL (Liu et al., 2025) derives implicit step rewards from trajectory preferences. MISE further adjusts hindsight self-evaluations against environment success rates (Yao et al., 2026). Related conformal methods calibrate orchestration decisions or certify retained reasoning prefixes (Si et al., 2026; Cheung et al., 2026). In contrast, CoRLA applies teacher-label conformal intervals directly to noisy transition-reward inference, then uses the resulting intervals to constrain return-anchored potential learning and select OPD supervision, connecting local progress, long-horizon credit, and token-level guidance in one policy update.

5 Conclusion

This work introduced CoRLA, a reinforcement learning framework that converts noisy next-state feedback into local, long-horizon, and token-level supervision for LLM agents. It constructs teacher-label conformal reward intervals, learns a return-anchored potential for temporal credit, and selectively applies hindsight OPD. Across terminal operation, software repair, and tool-assisted mathematics, CoRLA achieved higher scores than OpenClaw-RL. Performance gains tended to increase with trajectory length, and learned credits aligned more closely with counterfactual return changes than local process rewards. Because calibration targets teacher labels and coverage is evaluated empirically under adaptive online RL, reliability remains bounded by teacher quality and distribution shift.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

We used generative AI tools primarily to revise and polish the language of the manuscript and to obtain feedback on its quality, including suggestions for improving clarity, organization, methodological descriptions, and experimental reporting. The authors are responsible for evaluating these suggestions and take full responsibility for the accuracy, originality, and integrity of the final manuscript, including its methods, results, and conclusions. The frozen LLM teacher used to generate process labels and hindsight hints is a component of CoRLA, described in Section 2 and Appendices B and D.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

This work studies LLM-agent training on existing benchmarks for terminal interaction, software repair, and tool-assisted mathematics. Teacher-generated process labels and hints may inherit errors or biases from the underlying model, and uncertainty estimates do not guarantee safe or unbiased behavior. More capable tool-using agents can also produce unintended actions or be misused. Deployment should therefore include task-appropriate safeguards, such as restricted tool permissions, isolated execution environments, and human oversight for consequential actions. The reported benchmark improvements should be interpreted within the evaluated settings and do not establish readiness for unrestricted deployment.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

Our code is available at `https://anonymous.4open.science/r/CoRLA-BF10`. Section 2 specifies the model components, learning objectives, and update rules, and Appendix A provides the overall training algorithm. Section 3 describes the benchmarks, baselines, and evaluation metrics. Appendix B documents model initialization, optimization, rollout budgets, and calibration settings. Appendix C describes the protocols for reward-noise robustness, long-horizon credit assignment, and uncertainty-aware gating, with proposed settings explicitly identified as such. Appendix D provides the prompts for process reward evaluation and hint extraction. These descriptions support examination and reimplementation of the method and its evaluation procedures.

REFERENCES

Anthropic. Claude opus 4.1, August 2025a. URL `https://www.anthropic.com/news/claude-opus-4-1`. Accessed: 2026-09-11.

Anthropic. Introducing claude sonnet 4.5, September 2025b. URL `https://www.anthropic.com/news/claude-sonnet-4-5`. Accessed: 2026-09-11.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. Conformal prediction beyond exchangeability. *The Annals of Statistics*, 51(2):816–845, 2023.

Matt Y Cheung, Ashok Veeraraghavan, Hanjie Chen, and Guha Balakrishnan. Conformal certification of reasoning trace prefixes. *arXiv preprint arXiv:2605.30085*, 2026.

Shengda Fan, Xuyan Ye, Yupeng Huo, Zhi-Yuan Chen, Yiju Guo, Shenzhi Yang, Wenkai Yang, Shuqi Ye, Jingwen Chen, Haotian Chen, et al. Agentprocessbench: Diagnosing step-level process quality in tool-using agents. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 8823–8834, 2026.

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 *International Conference on Learning Representations*, volume 2026, pp. 37909–37926, 2026a.

Xiao Feng, Bo Han, Zhanke Zhou, Jiaqi Fan, Jiangchao Yao, Ka Ho Li, Dahai Yu, and Michael Kwok-Po Ng. Rewardflow: Topology-aware reward propagation on state graphs for agentic rl with large language models. *arXiv preprint arXiv:2603.18859*, 2026b.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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. *arXiv preprint arXiv:2503.09516*, 2025.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523): 1094–1111, 2018.

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 *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Online process reward learning for agentic reinforcement learning. *arXiv e-prints*, pp. arXiv–2509, 2025.

MAA. American Invitational Mathematics Examination - AIME, 2024.

MAA. American Invitational Mathematics Examination - AIME, 2025.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Moonshot AI. Kimi k2: Open agentic intelligence, 2025. URL https://moonshotai.github.io/Kimi-K2/. Accessed: 2026-09-11.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

OpenAI. Introducing SWE-bench Verified. Benchmark release, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

OpenAI. Introducing gpt-5, August 2025a. URL https://openai.com/index/introducing-gpt-5/. Accessed: 2026-09-11.

OpenAI. Introducing gpt-5.2, December 2025b. URL https://openai.com/index/introducing-gpt-5-2/. Accessed: 2026-09-11.

OpenAI. Introducing gpt-oss, August 2025c. URL https://openai.com/index/introducing-gpt-oss/. Accessed: 2026-09-11.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553, 2026.

Qwen Team. Qwen3-coder: Agentic coding in the world, July 2025. URL https://qwen.ai/blog?id=qwen3-coder. Accessed: 2026-09-11.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 60808–60838, 2025.

Qijia Shen, Zhiqi Huang, Vamsidhar Kamanuru, Aznaur Aliev, Jay Rainton, Ahmed Awelkair, Zhichen Zeng, Jiajun Li, Shi Dong, Yueming Yuan, et al. Seta: Scaling environments for terminal agents. *arXiv preprint arXiv:2607.10891*, 2026.

Wenwen Si, Sooyong Jang, Insup Lee, and Osbert Bastani. Conformal constrained policy optimization for cost-effective llm agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25446–25453, 2026.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

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, 2024.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195, 2026.

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

Jiashu Yao, Heyan Huang, Zeming Liu, and Yuhang Guo. Utilizing and calibrating hindsight process rewards via reinforcement with mutual information self-evaluation. *arXiv preprint arXiv:2604.11611*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

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

Zilin Zhu, Chengxing Xie, Xin Lv, et al. slime: An llm post-training framework for rl scaling. `https://github.com/THUDM/slime`, 2025. GitHub repository. Corresponding author: Xin Lv.

A Overall Training Algorithm

Algorithm 1 summarizes the alternating training procedure of CoRLA. After bootstrapping the judge and initializing calibration, each round collects trajectories with a frozen copy of the current policy. We first reserve random calibration probes, then query uncertain transitions from the remaining candidates to refresh the judge. The updated judge rescores the probe buffer, and the resulting intervals are cached. Potential learning updates its own adapter and head while keeping the judge fixed. The updated potential produces uncertainty-gated credits, which are propagated backward to form action-level advantages. Selective OPD supplies token-level targets from transitions with actionable hints, using the frozen rollout policy for scoring. Finally, the policy adapter is updated with both forms of supervision and KL regularization toward the initial policy, while all other trainable parameters, cached targets, and reference probabilities remain fixed.

Algorithm 1 CoRLA: alternating online training

Input: Frozen backbone F_{θ_0}, language-model head, and LLM teacher J_{LLM}; initial adapters $\Delta_\pi, \Delta_J, \Delta_\Phi$ and heads g_η, g_ψ; task distribution $\mathcal{D}$; training hyperparameters.
Output: Trained policy π_{Δ_π}.

Freeze the initial-policy reference $\pi_{\mathrm{init}} \leftarrow \pi_{\Delta_\pi}$ for all rounds.
Collect initial-policy transitions and teacher labels; randomly reserve a calibration subset and use the remaining pairs as judge training data $\mathcal{D}_J^{(0)}$.
Bootstrap only (Δ_J, η) on $\mathcal{D}_J^{(0)}$ with $\mathcal{L}_{\mathrm{NS}}$ (Eq. 2).
Initialize $\mathcal{B}^{\mathrm{judge}} \leftarrow \mathcal{D}_J^{(0)}$ and $\mathcal{B}^{\mathrm{teach}}$ with the held-out subset; compute q^{teach} (Eq. 3).
for each online round k **do**
  Freeze $\pi_{\mathrm{old}} \leftarrow \pi_{\Delta_\pi}$; collect trajectories $\mathcal{D}_\tau$, new transitions $\mathcal{D}_{\mathrm{new}}$, and environment rewards; compute returns G_t (Eq. 5).
  Draw $\mathcal{S}^{\mathrm{cal}} \sim \mathrm{Uniform}_{M_{\mathrm{cal}}}(\mathcal{D}_{\mathrm{new}})$ without replacement, before uncertainty ranking.
  Score $\mathcal{D}_{\mathrm{new}} \setminus \mathcal{S}^{\mathrm{cal}}$ with the judge; select the $K \le N - M_{\mathrm{cal}}$ largest scales as $\mathcal{S}^{\mathrm{judge}}$.
  Query J_{LLM} for both sets; form $\mathcal{D}_J^{(k)}$ and the labeled calibration probes.
  Add $\mathcal{D}_J^{(k)}$ to $\mathcal{B}^{\mathrm{judge}}$; update only (Δ_J, η) with $\mathcal{L}_{\mathrm{NS}}$, mixing new labels and uniformly sampled historical labels from this replay buffer.
  Add calibration probes to $\mathcal{B}^{\mathrm{teach}}$ and retain recent probes; exclude these pairs from judge regression and replay.
  Rescore $\mathcal{B}^{\mathrm{teach}}$ with the updated judge and recompute q^{teach} (Eq. 3).
  Cache judge means, interval endpoints, half-widths b_t, and gates ω_t with stop-gradient (Eqs. 4 and 8).
  Update only (Δ_Φ, ψ) with $\mathcal{L}_\Phi$ (Eq. 7), fixing cached intervals, (Δ_J, η), and Δ_π; apply the soft interval penalty only for $t < T$.
  Recompute Φ_t; set $\Phi_{T+1} = 0$ at true termination and retain bootstrap potentials at nonterminal truncations.
  Cache credits c_t and advantages $\widehat{A}_t$ from the updated potential (Eqs. 9–10).
  Initialize $m_t^{\mathrm{OPD}} \leftarrow 0$ and $\widehat{A}_{t,j}^{\mathrm{OPD}} \leftarrow 0$ for all action tokens.
  for each transition with cached $b_t \le \tau_{\mathrm{OPD}}$ **do**
    Extract an actionable hint g_t with J_{LLM}, or set $g_t = \varnothing$ if none is available.
    if $g_t \neq \varnothing$ **then**
      Set $m_t^{\mathrm{OPD}} \leftarrow 1$ and $h_t^+ \leftarrow h_t \oplus g_t$.
      Use π_{old} to score the sampled action under h_t^+ and h_t; cache clipped token targets $\widehat{A}_{t,j}^{\mathrm{OPD}}$ (Eq. 12).
    end if
  end for
  Update only Δ_π with $\mathcal{L}_\pi$ (Eq. 14) and KL regularization, using agent-generated action positions only; fix judge and potential adapters and heads, reference policies, masks, and cached targets.
end for

B Training and Implementation Details

Following the general-agent training setup of OpenClaw-RL (Wang et al., 2026), we initialize the terminal and SWE policies with Qwen3-8B and Qwen3-4B (Yang et al., 2025), respectively, and the tool-call policy with Qwen3-4B-SFT, the supervised fine-tuned checkpoint provided by slime (Zhu et al., 2025). A frozen Qwen3-4B supplies process labels and hindsight hints; the frozen rollout policy supplies the hint-conditioned OPD token probabilities. The backbone and language-model head are frozen. The policy and critic use separate LoRA adapters with rank $r = 16$, $\alpha_{\mathrm{LoRA}} = 32$ (effective scale $\alpha_{\mathrm{LoRA}}/r = 2$), and dropout 0.05, targeting all attention and feed-forward projection layers. The judge head g_η and potential head g_ψ are two-layer MLPs with 1024 hidden units and SiLU activations; their output dimensions are two and one, respectively. We train the terminal, SWE, and tool-call agents on SETA RL data (Shen et al., 2026), SWE-Gym (Pan et al., 2024), and DAPO RL data (Yu et al., 2026; 2025), respectively.

We use AdamW with a learning rate of 10^{-6} for all trainable parameters, $(\beta_1, \beta_2) = (0.9, 0.95)$, numerical epsilon 10^{-8}, zero weight decay, and global gradient-norm clipping at 1.0. Training runs for 500 online rounds, with 20 rounds of linear learning-rate warm-up followed by a constant rate. Each round collects a total of 8, 16, or 32 trajectories for SWE, terminal, or tool-call training, respectively, and makes one policy update using the entire rollout batch. We add forward KL regularization relative to the frozen initial policy, with coefficient 0.01, to Eq. (14); the KL is evaluated over the token distribution at agent-generated positions with the same trajectory averaging and temporal weighting as the policy loss. The lower and upper clipping parameters are 0.2 and 0.28, giving the probability-ratio interval $[0.8, 1.28]$. Rollouts use temperature 1.0 and top-$p = 1.0$. The context limit is $32{,}768$ tokens, with at most $4{,}096$ generated tokens per turn and $16{,}384$ per trajectory. The maximum interaction lengths are 10, 20, and 10 steps for terminal, SWE, and tool-call tasks, respectively.

Before fitting, we randomly reserve $1{,}024$ calibration transitions and bootstrap the judge on the remaining $4{,}096$ labeled transitions for 200 updates. From each online batch of N transitions, we first sample $M_{\mathrm{cal}} = \min(64, \lceil N/2 \rceil)$ calibration probes uniformly without replacement, then select $K = \min(64, N - M_{\mathrm{cal}})$ remaining transitions with the largest predicted scales for judge queries. Calibration uses $\alpha_{\mathrm{teach}} = 0.1$ and retains at most $2{,}048$ probes from the latest 5 rounds. Judge replay retains $10{,}000$ transitions and samples new and historical labels in a $1{:}1$ ratio; calibration probes are excluded. Each round takes five judge updates on replay, rescores the full probe buffer to recalibrate intervals, and then takes five potential updates on current rollouts, using 128-transition mini-batches. We set $\delta = 1$, $\sigma_{\min} = 10^{-3}$, $\gamma = 0.99$, $\lambda_A = 0.95$, $\lambda_{\mathrm{CP}} = 0.1$, $\beta_\Phi = 1$, and $\mathcal{R} = 1$. For OPD, $\tau_{\mathrm{OPD}} = 0.5$ is fixed throughout training, with $C_{\mathrm{OPD}} = 2$ and $\lambda_{\mathrm{OPD}} = 0.1$.

C Experimental Implementation Details

This section specifies the protocols for reward-noise robustness, long-horizon credit assignment, and uncertainty-aware gating in Sections 3.4–3.6.

C.1 Reward Noise Robustness

Data partition and corruption bookkeeping. AgentProcessBench has clean step labels $y_i \in \mathcal{Y} = \{-1, 0, +1\}$ and a training/calibration/test split of $8{:}1{:}1$. The proposed partition assigns complete trajectories, rather than individual steps, to disjoint subsets with a fixed random seed of 42; step counts can therefore differ from this ratio. Task-level duplicates should remain in the same subset. Clean labels are retained as reference annotations, and $\widetilde{y}_i$ denotes a corrupted label. The corruption indicator is $z_i = \mathbf{1}\{\widetilde{y}_i \neq y_i\}$. Training uses corrupted labels; LS-NoiseCal uses a separately corrupted calibration copy, whereas LS-OracleCal uses its clean counterpart. Accuracy and coverage are evaluated against clean test labels. All variants should use the same split and corruption realizations. We employ the following four types of noises:

- **Symmetric noise.** For each eligible step, sample $B_i \sim \mathrm{Bernoulli}(\rho)$. If $B_i = 0$, retain y_i; otherwise sample $\widetilde{y}_i$ uniformly from $\mathcal{Y} \setminus \{y_i\}$. Thus each incorrect class has probability $\rho/2$, and the clean class has probability $1 - \rho$.

- **Asymmetric noise.** Use the same Bernoulli corruption trigger, but restrict changes to adjacent classes: $-1 \mapsto 0$, $+1 \mapsto 0$, and $0 \mapsto -1$ or $+1$ with equal probability. No direct flip between -1 and $+1$ is permitted. The proposed rate for both symmetric and asymmetric noise is $\rho = 0.2$.
- **Heteroscedastic noise.** Repeatedly score the same transition with a frozen judge and an unchanged scoring prompt. We use five independent calls at temperature 0.7 and top-$p = 1.0$, varying the decoding seed. A step is ambiguous if at least two valid returned labels disagree. Apply the symmetric corruption rule with probability $\rho = 0.2$ only to ambiguous steps; retain all other labels. Consequently, 0.2 is the conditional corruption probability, and the overall rate depends on the fraction of ambiguous steps. The repeated judge identifies ambiguity only; the clean dataset annotation remains the reference label.
- **Temporal credit-shift noise.** Let $e = \min\{t : y_t = -1\}$ be the first error in a trajectory. A proposed one-step delay exchanges y_e with y_{e+1} when $e < T$ and $y_{e+1} \neq -1$, placing the first negative label at $e + 1$ while preserving label counts. Trajectories without an error, with an error only at the final step, or with another negative label immediately after e remain unchanged.

Predictors and interval construction. Point RM uses only a reward location $\mu_{\mathrm{pt}}(x)$. The LS variants predict a location $\mu(x)$ and positive scale $\sigma(x)$ using the parameterization in Section 2.1. For the uncalibrated LS RM, the proposed interval is

$$I_{\mathrm{LS}}(x) = [\mu(x) - c\sigma(x),\ \mu(x) + c\sigma(x)], \qquad c = 1. \tag{16}$$

This is a fixed-scale baseline, with no nominal coverage guarantee; σ is not assumed to be a Gaussian standard deviation. For either conformalized variant, compute $s_i = |y_i^{\mathrm{cal}} - \mu(x_i)|/\sigma(x_i)$ on the calibration subset and let q be the $\lceil (n_{\mathrm{cal}} + 1)(1 - \alpha) \rceil$-th smallest score, using $q = +\infty$ if that rank exceeds n_{cal}. The interval is

$$I_{\mathrm{CP}}(x) = [\mu(x) - q\sigma(x), \mu(x) + q\sigma(x)]. \tag{17}$$

The proposed offline miscoverage level is $\alpha = 0.1$. LS-NoiseCal uses $y_i^{\mathrm{cal}} = \widetilde{y}_i$, and LS-OracleCal uses $y_i^{\mathrm{cal}} = y_i$. These offline quantiles are separate from the online teacher-buffer quantile. Calibration to noisy labels does not guarantee coverage of clean labels.

Discrete label prediction and coverage. A proposed common decision rule maps each model's location to its nearest class, resolving midpoint ties in favor of zero:

$$\widehat{y}(x) = \begin{cases} -1, & \mu(x) < -0.5, \\ 0, & -0.5 \le \mu(x) \le 0.5, \\ +1, & \mu(x) > 0.5. \end{cases} \tag{18}$$

For Point RM, substitute μ_{pt} for μ. StepAcc is $100 n_{\mathrm{test}}^{-1} \sum_i \mathbf{1}\{\widehat{y}(x_i) = y_i\}$. Coverage is $100 n_{\mathrm{test}}^{-1} \sum_i \mathbf{1}\{y_i \in I(x_i)\}$, with inclusive endpoints and $I = I_{\mathrm{LS}}$ or I_{CP} as appropriate. Point RM has no interval, so its coverage is not reported. Intervals remain untruncated; intersecting them with $[-1, 1]$ would not change coverage of these labels, but would change their widths.

C.2 Long-Horizon Credit Assignment

Short, medium, and long task groups. Groups are defined separately for each benchmark from base-model trajectory lengths and then held fixed across evaluated policies. Length counts agent actions, rather than tokens or environment messages. The proposed deterministic implementation sorts the n tasks by $(T_i, \mathrm{task_id}_i)$, using task identifiers to break equal-length ties. With $k_1 = \lfloor n/3 \rfloor$ and $k_2 = \lfloor 2n/3 \rfloor$, ranks 1 through k_1 form the short group, $k_1 + 1$ through k_2 the medium group, and $k_2 + 1$ through n the long group. These are empirical rank tertiles, approximately the 33rd and 67th percentiles. Group sizes differ by at most one, and equal-length tasks may fall in different groups at a boundary.

Counterfactual action sampling. A proposed budget is $M = 8$ replacement actions per evaluated step. Restore the environment and interaction history immediately before a_t, and independently

sample each replacement from a fixed proposal policy. Both replacements and continuations use temperature 1.0 and top-$p = 1.0$. A common frozen base-model policy for both action proposals and all subsequent continuations provides a reproducible comparison across credit estimators. Sampling is with replacement: duplicate actions, including a sample identical to the original, are retained rather than selecting alternatives by their outcomes. Each continuation starts from a fresh copy of the same pre-action state, and all credit estimators are evaluated on the same interventions and returns.

C.3 Uncertainty-Aware Gating

A fixed hard-filter threshold. The hard-filter configuration uses $\tau_g = 0.5$, fixed across tasks and training rounds. This matches the existing OPD half-width threshold $\tau_{\mathrm{OPD}} = 0.5$ because the reward-range radius is $\mathcal{R} = 1$:

$$\omega_t^{\mathrm{hard}} = \mathbf{1}\{b_t/\mathcal{R} \le 0.5\} = \mathbf{1}\{b_t \le \tau_{\mathrm{OPD}}\}, \qquad \mathcal{R} = 1. \tag{19}$$

Thus both mechanisms apply the same width cutoff; OPD additionally requires a nonempty actionable hint. Hard filtering controls the potential-difference term, whereas the OPD mask selects token-level distillation supervision. Equal thresholds therefore do not make the two mechanisms identical. For a general reward radius, matching the cutoffs requires $\tau_g = \tau_{\mathrm{OPD}}/\mathcal{R}$.

Comparison with the soft gate. CP Soft Gate uses $1 - \mathrm{clip}(b_t/\mathcal{R}, 0, 1)$ and has no hard threshold τ_g. At $b_t = 0.5$ it retains weight 0.5, whereas Hard Filtering Gate retains weight 1; above 0.5, hard filtering sets the correction to zero while the soft gate decreases continuously until $b_t = 1$. Environment rewards and the separately defined OPD term remain active under their existing rules.

D Prompt Templates

We use the prompts in OpenClaw-RL (Wang et al., 2026) for process reward evaluation and hint extraction.

Prompt for Process Reward Judge

```
``` <|im_start|>system

You are a process reward model (PRM) evaluating an AI assistant.
You will see the assistant's output and the subsequent next state.
Your task: decide whether the assistant's output **successfully
    fulfilled** the user's intent at that step, using the next state as
     evidence.

Understanding the next state's role
- role='user': A reply from the user.
- role='tool': The return value of a tool the assistant invoked. This
    content was NOT available before the assistant's action -- it
    exists BECAUSE the assistant called the tool. A successful, non-
    error tool output means the assistant's action worked correctly and
     should be scored positively.

Scoring rules
- \boxed{1} (good): The next state shows the task progressed as
    expected -- e.g. the user moves on, says thanks, the environment
    confirms success, or a tool returns a successful, non-error result.
- \boxed{-1} (bad): The next state signals the assistant's output was
    wrong, incomplete, or unwanted. **Key negative signals include:**
   * The user asks the assistant to **redo, retry, or repeat** the
    same action (``do it again'', ``try again'', ``one more time'').
   * The user requests a **correction or modification** to what the
    assistant just did (``change X to Y'', ``no, I meant ...'', ``not
    that, ...'', ``please fix ...'').
   * The user **rephrases or restates** the same request, implying the
     assistant did not understand or execute it correctly.
```
```

```
   * The environment returns an **error, failure, or unexpected result
    ** caused by the assistant's action.
- \boxed{0} (neutral): The next state is ambiguous -- e.g. the user
    gives an unrelated follow-up that neither confirms nor denies
    success, or there is insufficient information to judge.

## Important
A change request IS negative feedback -- it means the previous output
    did not meet  the user's need. Do NOT treat it as a neutral new
    instruction.

Think step-by-step, then give your final score inside \boxed{}.
<|im_end|>
<|im_start|>user
## Assistant output
{{response_text}}

## Next state [role: {{next_state_role}}]
{{next_state_text}}

First, classify the next state: is it (a) positive progression, (b) a
    correction / redo / change request, or (c) ambiguous? Then assign \
    boxed{1}, \boxed{-1}, or \boxed{0}.
<|im_end|>
'''
```

Prompt for Hint Extraction

```
``` <|im_start|>system

You are a process reward model used for hindsight hint extraction.
You are given:
1) The assistant response at turn t.
2) The next state at turn t+1, along with its **role**.

Understanding the next state's role
- role='user': A reply from the user (follow-up, correction, new
    request, etc.).
- role='tool': The return value of a tool the assistant invoked. This
    content was NOT available before the assistant's action -- it
    exists BECAUSE the assistant called the tool. A successful, non-
    error tool output generally means the assistant's action was
    appropriate; do NOT treat it as information the assistant should
    have already known.

Your goal is to decide whether the next state reveals useful hindsight
     information that could have helped improve the assistant response
    at turn t.

Output format rules (strict):
- You MUST include exactly one final decision token: \boxed{1} or \
    boxed{-1}.
- If and only if decision is \boxed{1}, provide a concise, information-
    dense hint in 1-3 sentences, wrapped between [HINT_START] and [
    HINT_END].
- If decision is \boxed{-1}, do not provide a hint block.
- Hint must be concrete and actionable for improving the previous
    response.
<|im_end|>
<|im_start|>user
Assistant response (turn t)
{{response_text}}
```
```

```
## Next state (turn t+1) [role: {{next_state_role}}]
{{next_state_text}}

Now output your decision and (if positive) the hint in the required
    format.
<|im_end|>
'''
```