When Credit Turns Against Tools: Transition-Aware Credit Assignment in Agentic Reinforcement Learning

Xiangyu Lu[1] **Zhanke Zhou**[1] **Jiazhe Ning**[1,2] **Chentao Cao**[1]
Jiangchao Yao[3] **Bo Han**[1†]

[1]TMLR Group, Department of Computer Science, Hong Kong Baptist University
[2]Peking University [3]Cooperative Medianet Innovation Center, Shanghai Jiao Tong University

Abstract

Agentic reinforcement learning with verifiable rewards trains language models through multi-turn tool interactions, yet their reasoning performance and tool-calling ability can decline during training. We trace this degradation to three coupled mechanisms: (1) *tool-token imbalance* produces negative overall tool credit when incorrect trajectories contain more tool tokens per turn than correct ones; (2) *aggregation amplification* can further strengthen this negative signal as token-mean aggregation gives longer incorrect trajectories greater weight in policy updates; and (3) *failure-driven retries* prolong trajectories as models tend to call tools again after execution errors, extending potential negative credit to recovery attempts. To counter this suppression, we propose **Transition-Aware Credit Assignment (TACA)**, which targets the transition from reasoning to tool use through supervision of both call initiation and generation. Specifically, TACA uses *Advantage Projection* to enforce a nonnegative mean advantage on tool-call tokens within each prompt group through a minimal uniform shift. Additionally, TACA guides tool-use decisions with *Decision Contrast*, comparing rewards for tool-assisted and direct answers to the same prompt. Through *Supervised Replay*, it reinforces tool-call generation using calls from previously sampled correct trajectories. Across models and tasks, TACA improves performance and sustains more reliable tool interaction during training, raising mean accuracy over MT-GRPO on six reasoning benchmarks by 4.61 percentage points for Qwen2.5-1.5B and 3.61 for Qwen2.5-3B. Our code is available at `https://github.com/tmlr-group/TACA`.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has improved reasoning in language models by using final-answer correctness as a terminal reward [2, 39, 34, 10, 37, 56, 54, 14, 49, 47]. Agentic RLVR extends this approach to settings with multi-turn tool interactions, where models use the observations returned by tools to guide subsequent reasoning [51, 18, 44, 5, 29, 58, 33, 36]. Multi-turn Group Relative Policy Optimization (MT-GRPO) [34, 6, 26], a widely used approach to agentic RLVR, assigns each trajectory's terminal reward to every model-generated turn. Within a trajectory, the tokens that invoke a tool and those that reason from its output therefore share the same advantage.

Despite these advances, training instability has been repeatedly documented in agentic reinforcement learning systems [51, 44, 42, 4, 50, 9, 55, 3]. Specifically, reasoning accuracy can improve early in training and then fall below the non-agentic baseline. During this decline, tool-use rates can remain broadly stable even as a smaller fraction of calls execute successfully. The model observes whether a call succeeds within the current turn, but MT-GRPO assigns credit to that call based on the final

†Correspondence to Bo Han (bhanml@comp.hkbu.edu.hk).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

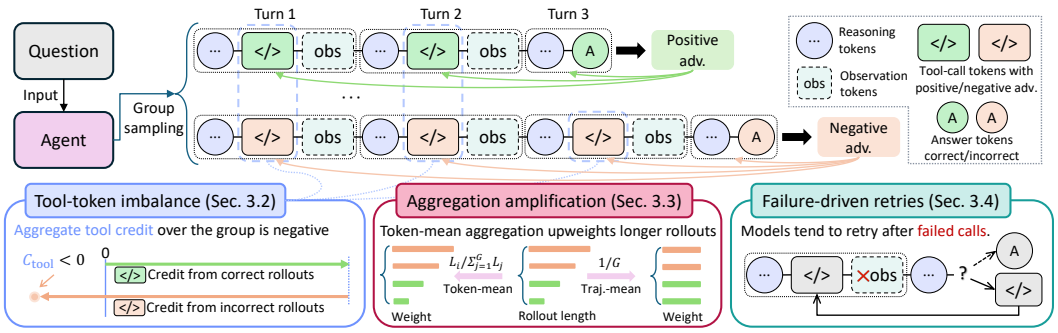

Figure 1: Coupled mechanisms of training degradation. Terminal supervision can yield negative aggregate tool credit. Token-mean aggregation can further amplify this signal by upweighting longer incorrect rollouts, while retries prolong these rollouts and extend negative credit to recovery attempts.

answer, potentially several turns later. Under this credit assignment, the same successful call can be reinforced in a trajectory that ends correctly and discouraged in one that ends incorrectly. This raises a natural question: *how do the positive and negative tool-call credits combine across trajectories, and can they explain why tool execution and reasoning performance deteriorate with continued training?*

We therefore analyze how positive and negative advantages combine across the tokens that form tool calls in MT-GRPO (Figure 1). We show that when incorrect trajectories contain more tool-call tokens per generated turn than correct ones, this *tool-token imbalance* produces negative overall credit for tool calls, even when correct trajectories constitute a majority. Additionally, *aggregation amplification* can strengthen this negative signal, as token-mean aggregation gives negative credit from longer incorrect trajectories greater weight in policy updates than it would receive under equal trajectory weighting. Finally, models tend to retry on the next turn after a tool call fails. These *failure-driven retries* occur more frequently in incorrect trajectories, where each additional tool-calling turn extends negative credit to the recovery attempt and gives the trajectory greater weight in the policy update.

Motivated by these findings, we propose **T**ransition-**A**ware **C**redit **A**ssignment (**TACA**), which augments MT-GRPO with targeted supervision for the transition from reasoning to tool use. TACA trains both the decision to seek external feedback and the generation of the call that determines what the model observes next. Specifically, *Advantage Projection* offsets negative mean advantage on tool-call tokens while preserving reward-dependent advantage differences within each prompt group. *Decision Contrast* guides tool-use decisions by comparing rewards between tool-using and direct-answer trajectories sampled for the same prompt. *Supervised Replay* reuses previously sampled tool calls as supervision targets across policy updates. Together, these components train the transition from reasoning to tool interaction using evidence beyond the final outcome of an individual trajectory.

We evaluate TACA on four Qwen and Llama models across mathematical reasoning, coding, and decision-making. TACA achieves the best average accuracy in the Qwen2.5-1.5B and 3B baseline comparisons and improves over turn-level GRPO on Qwen2.5-7B and Llama-3.2-1B. With Qwen2.5-1.5B, it raises the TACO unit-test score from 41.98 to 53.70 and WebShop success rate from 56.8% to 67.2%. These gains accompany more reliable tool execution during training, with ablations supporting the complementary roles of advantage adjustments and replay supervision.

The main contributions of this work are:

- We identify negative aggregate tool credit as a mechanism underlying training degradation in multi-turn GRPO. Our analysis links its emergence to tool-token imbalance, with length weighting amplifying its influence and failure-driven retries extending negative supervision (Sec. 3).
- We propose TACA, a transition-aware objective that jointly optimizes tool-use decisions and tool-call generation by correcting negative aggregate tool credit, guiding tool-use decisions through reward contrasts, and maintaining tool-use capabilities through replay supervision (Sec. 4).
- We demonstrate that TACA consistently improves performance over existing RL methods across model families and tasks spanning mathematical reasoning, coding, and decision-making. TACA also sustains reasoning performance and tool-execution reliability as training progresses (Sec. 5).

2 Preliminaries

Multi-turn rollouts and advantages. In multi-turn interaction, each model-generated turn may contain reasoning, a tool call, or an answer; returned observations enter the context for subsequent turns. For each prompt x, a fixed rollout policy $\pi_{\theta_{\text{old}}}$ samples G trajectories $\{\tau_i\}_{i=1}^{G}$ where trajectory τ_i contains S_i generated turns and receives a bounded scalar terminal reward R_i.

Group Relative Policy Optimization (GRPO) [34] computes advantages by standardizing rewards within each prompt's sample group. We study multi-turn GRPO (MT-GRPO) with turn-level reward normalization [26, 6, 59, 30, 17, 43]: each trajectory reward R_i is replicated across its S_i turns before normalization. With $M = \sum_i S_i$ total turns, the reward mean and variance are

$$\mu_{\text{turn}} = \frac{1}{M}\sum_i S_i R_i, \qquad \sigma^2_{\text{turn}} = \frac{1}{M}\sum_i S_i (R_i - \mu_{\text{turn}})^2. \tag{1}$$

Every generated token in τ_i then receives the same advantage $A_i^{\text{turn}} = (R_i - \mu_{\text{turn}})/\sigma_{\text{turn}}$.

Token-mean policy objective. Let $\ell_{i,h}$ denote the number of generated tokens scored by the objective in turn h of trajectory i, and $N = \sum_i \sum_{h=1}^{S_i} \ell_{i,h}$ their total count. Observations provide context but are excluded from the objective. We use token-mean aggregation [54, 23, 18, 51, 48]. The current policy π_θ maximizes the objective averaged over these N tokens:

$$\mathcal{J}_{\text{MT-GRPO}}(\pi_\theta) \triangleq \frac{1}{N}\sum_{i=1}^{G}\sum_{h=1}^{S_i}\sum_{t=1}^{\ell_{i,h}} \Big[\phi(\rho_{i,h,t}, A_i^{\text{turn}}) - \beta\, \mathbb{D}_{\text{KL}}\big[\pi_\theta \,\big\|\, \pi_{\text{ref}}\big]_{i,h,t}\Big], \tag{2}$$

Here, $\rho_{i,h,t}$ is the token-level importance sampling ratio of the current policy relative to the rollout policy. The clipped surrogate $\phi(\rho, a) = \min\{\rho a, \text{clip}(\rho, 1-\varepsilon, 1+\varepsilon)a\}$ uses advantage a and clipping threshold ε. The KL penalty has weight β and fixed reference policy π_{ref}.

3 Understanding Training Degradation in Multi-Turn GRPO

We analyze how terminal-reward supervision, reward normalization, and objective aggregation determine credit on tool interactions. We first derive the scalar credit and trajectory-weighting relationships, then examine how retry behavior contributes to these effects. Details are in Appendix F.

3.1 Training Degradation and Tool Interaction

Training instability in multi-turn agentic RL has been documented across recent systems [51, 18, 44, 42, 4, 50, 9, 55, 3, 11]. We reproduce this degradation in a widely adopted tool-integrated reasoning setting, training Qwen2.5-3B-Instruct on DAPO-Math with MT-GRPO. After an initial improvement, validation accuracy declines and falls below the non-agentic reference (Figure 2a). Meanwhile, tool-execution success declines while tool-use rates remain broadly stable (Figure 2b), and incorrect trajectories contain more tool calls on average than correct ones (Figure 2c). These observations suggest a close connection between training degradation and the quality of tool interaction.

3.2 Terminal-Reward Supervision and Negative Tool Credit

MT-GRPO assigns each trajectory's terminal-reward advantage to all its scored tokens. Turn-level normalization enforces $\sum_i S_i A_i^{\text{turn}} = 0$, balancing positive and negative advantage across generated turns. This balance can break on tool tokens: when incorrect trajectories contain more tool tokens per turn than correct ones, negative coefficients dominate the aggregate tool credit.

Consider a prompt group containing both correct ($R_i = 1$) and incorrect ($R_i = 0$) trajectories. Let m_i count the model-generated tokens forming tool calls in trajectory i. Let $\bar{m}^+_{\text{turn}}$ and $\bar{m}^-_{\text{turn}}$ denote the average number of tool tokens per generated turn in correct and incorrect trajectories, respectively.

Proposition 3.1 (Negative tool-credit condition)**.** *In the fully on-policy setting, the reward surrogate assigns coefficient A_i^{turn}/N to each tool token's log-probability gradient in trajectory i. Summing these coefficients gives*

$$C_{\text{tool}} \coloneqq \frac{1}{N}\sum_i m_i A_i^{\text{turn}} = \frac{M\sigma_{\text{turn}}}{N}\left(\bar{m}^+_{\text{turn}} - \bar{m}^-_{\text{turn}}\right), \tag{3}$$

where M denotes the total number of generated turns and N the total number of tokens included in the policy objective. Consequently, $C_{\text{tool}} < 0$ if and only if incorrect trajectories contain more tool tokens per generated turn than correct trajectories.

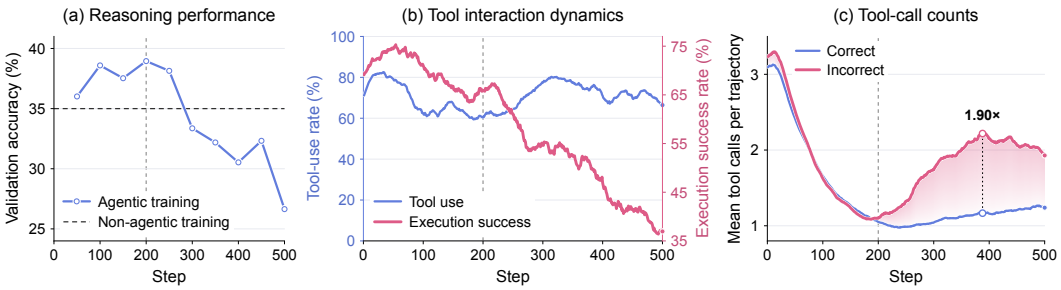

Figure 2: Training dynamics of MT-GRPO. (a) Validation accuracy peaks and then falls below the non-agentic reference. (b) Tool-execution success declines despite broadly stable tool-use rates. (c) Mean tool-call counts rise for incorrect trajectories and remain relatively stable for correct ones.

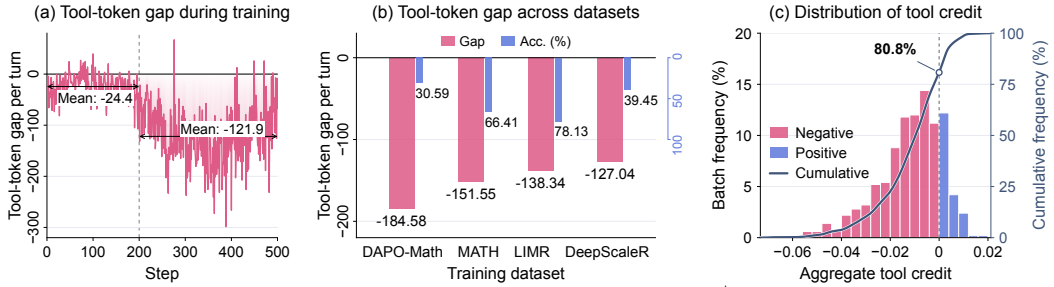

Figure 3: Negative credit on tool tokens. (a) The tool-token gap $\bar{m}^{+}_{\text{turn}} - \bar{m}^{-}_{\text{turn}}$ in tool tokens per turn becomes more negative during degradation. (b) At step 300, this gap remains negative across training datasets with different accuracies. (c) Aggregate tool credit C_{tool} is negative in most batches.

Proposition 3.1 shows that negative tool credit can arise even when correct trajectories constitute a majority. Within a mixed-outcome group, the difference in tool tokens per turn determines the sign of aggregate credit, while reward composition also affects its magnitude. Figure 3a,b characterize this imbalance during training and across datasets with different accuracies. Within such groups, terminal-reward supervision assigns the same negative advantage to every call in an incorrect trajectory, including calls that execute successfully before an error in subsequent reasoning. Tool-call credit is therefore tied to failures that may occur elsewhere in the trajectory. Under the condition in Proposition 3.1, these negative contributions collectively outweigh positive credit on tool tokens. Token-mean aggregation weights prompt-group credits by their total generated-token counts, and Figure 3c measures the resulting batch credit using advantages computed within each prompt group.

3.3 Length-Proportional Reweighting

Beyond the tool-token imbalance, aggregate tool credit also depends on trajectory weighting. We analyze the shift induced by token-mean aggregation relative to equal trajectory weighting.

Tool-credit aggregation. Let $\mathcal{B}$ index the trajectories in a training batch. For each $i \in \mathcal{B}$, let $L_i = \sum_{h=1}^{S_i} \ell_{i,h} > 0$ count the model-generated tokens and m_i count the tool-call tokens among them. With turn-level advantage A_i^{turn} computed within its prompt group, the trajectory contributes tool credit $(m_i/L_i)A_i^{\text{turn}}$ per generated token. Token-mean aggregation weights this contribution by trajectory length, whereas trajectory-mean aggregation weights trajectories equally:

$$C_{\text{tool}}^{\text{token}} = \sum_{i\in\mathcal{B}} \frac{L_i}{\sum_{j\in\mathcal{B}} L_j} \frac{m_i}{L_i} A_i^{\text{turn}}, \qquad C_{\text{tool}}^{\text{traj}} = \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \frac{m_i}{L_i} A_i^{\text{turn}}. \tag{4}$$

Aggregation-induced credit shift. Their difference isolates the effect of length weighting:

$$C_{\text{tool}}^{\text{token}} - C_{\text{tool}}^{\text{traj}} = \frac{\text{Cov}_{\mathcal{B}}\left(L_i, \frac{m_i}{L_i} A_i^{\text{turn}}\right)}{\mathbb{E}_{\mathcal{B}}[L_i]}, \tag{5}$$

where the empirical mean and covariance assign weight $1/|\mathcal{B}|$ to each trajectory. The covariance determines the direction of the credit shift: when longer trajectories tend to carry lower tool credit per generated token, token-mean aggregation shifts aggregate credit downward. Such a shift amplifies negative credit and reverses positive credit when its magnitude exceeds $C_{\text{tool}}^{\text{traj}}$.

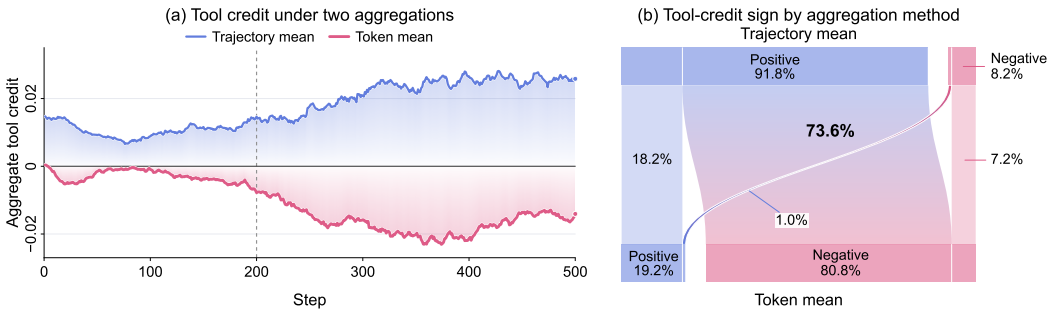

Figure 4: Length weighting and aggregate tool credit. (a) Token-mean aggregation shifts credit downward relative to trajectory-mean aggregation. (b) Length weighting turns positive tool credit negative in 73.6% of batches. This shifts the balance toward penalizing sampled tool calls.

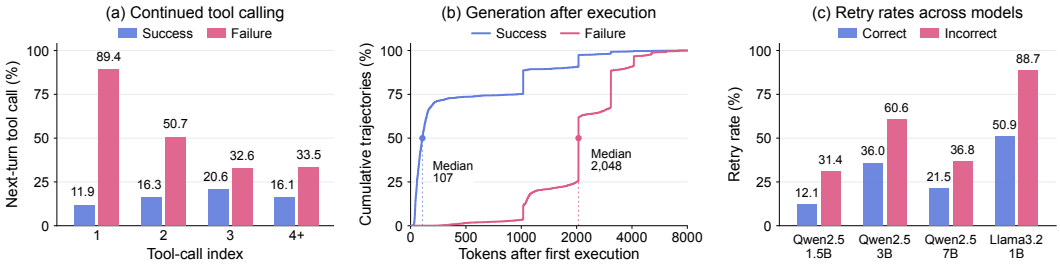

Figure 5: Retry behavior and continuation length. (a) A tool call in the next turn is more likely after failed than successful executions. (b) First-execution failure is associated with longer subsequent generation. (c) Incorrect trajectories have higher retry rates across all four models on MATH.

Figure 4a shows the downward credit shift induced by length weighting, while Figure 4b identifies batches whose tool credit changes from positive to negative. In these batches, negative tool-token coefficients outweigh positive coefficients in the reward surrogate, shifting the balance toward updates that penalize sampled tool calls. Length weighting thus compounds the suppression analyzed in Section 3.2: it increases the relative influence of negative supervision on tool interactions, including successful executions within ultimately incorrect trajectories.

3.4 Failure-Driven Retries

Failure-conditioned continuation. The trajectories underlying these credit imbalances are shaped by the model's response to execution errors. A retry adds another tool action and any accompanying reasoning before the terminal reward is assigned. Execution outcomes therefore influence which subsequent actions enter the training sample and how long generation continues. The higher next-turn tool-call rate after failed executions (Figure 5a) and longer continuation following an initial execution failure (Figure 5b) support this dependence of trajectory composition on intermediate outcomes.

Terminal supervision of recovery attempts. Each decision to retry expands the set of actions sharing the completed trajectory's terminal-reward advantage. Within a mixed-outcome prompt group, an ultimately incorrect trajectory assigns negative advantage to both the failed call and subsequent recovery attempts, including retries that execute successfully. Figure 5c shows higher retry rates in incorrect trajectories across models, connecting recovery behavior to the terminal outcomes that determine credit. The recovery behavior thus shapes subsequent supervision: in an unsuccessful trajectory, each additional recovery attempt adds tool actions receiving the same negative advantage.

4 Transition-Aware Credit Assignment

Motivated by Section 3, we propose Transition-Aware Credit Assignment (TACA), which augments MT-GRPO with token-level advantage corrections and auxiliary supervision for tool-call generation (Figure 6). Concretely, TACA offsets negative mean advantages on tool-call tokens (Advantage Projection), guides tool-use decisions by comparing rewards between tool-using and direct-answer trajectories sampled for the same prompt (Decision Contrast), and reinforces tool-call generation through an auxiliary likelihood objective on previously sampled calls (Supervised Replay).

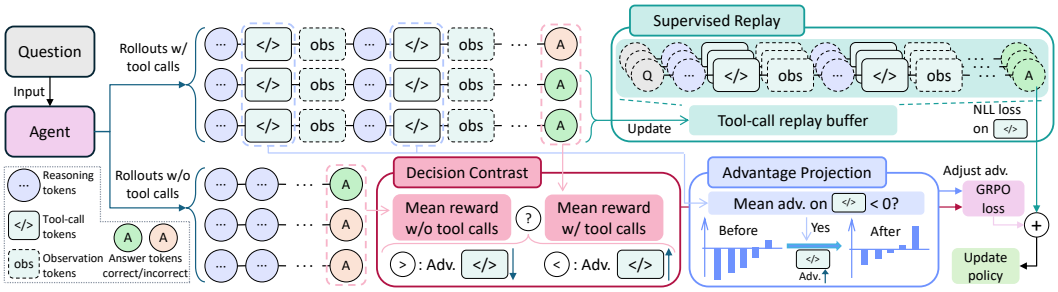

Figure 6: Overview of TACA. TACA adjusts tool-call and decision-token advantages using within-prompt reward statistics from tool-using and direct-answer rollouts. These adjustments and replay supervision from correct trajectories jointly guide tool-use decisions and tool-call generation.

4.1 Advantage Projection

We target the negative aggregate credit identified in Section 3 by applying a uniform advantage correction to tool-related tokens. For each prompt x, let $\mathcal{Z}$ denote the model-generated tokens forming tool calls across its sampled trajectories, and let A_i denote the GRPO advantage of trajectory i. We compute the mean advantage over these tokens and the corresponding correction:

$$\bar{A}_{x,\mathcal{Z}} = \frac{1}{|\mathcal{Z}|} \sum_{(i,h,t)\in\mathcal{Z}} A_i, \qquad b_{x,\mathcal{Z}} = \max\{-\bar{A}_{x,\mathcal{Z}}, 0\}, \tag{6}$$

where (i, h, t) indexes the trajectory, turn, and token position, and $|\mathcal{Z}|$ is the number of tool-call tokens in the set. Advantage Projection replaces A_i with $A_i + b_{x,\mathcal{Z}}$ exclusively at positions in $\mathcal{Z}$.

The correction changes the aggregate advantage over $\mathcal{Z}$ from $|\mathcal{Z}|\bar{A}_{x,\mathcal{Z}}$ to $|\mathcal{Z}|\max\{\bar{A}_{x,\mathcal{Z}}, 0\}$. Negative aggregate credit is therefore neutralized, while positive aggregate credit is preserved. The resulting vector is the Euclidean projection of the original tool-call advantages onto the halfspace of vectors with nonnegative mean. Within the same group, the uniform offset preserves the advantage difference $A_i - A_j$ between tool tokens from trajectories i and j. Advantage Projection therefore enforces nonnegative aggregate tool credit while retaining these reward-dependent differences.

4.2 Decision Contrast

We complement the tool-credit correction with a reward comparison that guides the choice between tool use and direct answering. For each prompt x, let $\mathcal{T}_x$ contain sampled trajectories that invoke at least one tool and $\mathcal{N}_x$ contain those that answer without tools. Let $\bar{R}_{\mathcal{T}_x}$ and $\bar{R}_{\mathcal{N}_x}$ denote their mean terminal rewards, counting each trajectory once. We define the reward contrast as

$$U_x = \frac{n_x}{n_x + \eta}\left(\bar{R}_{\mathcal{T}_x} - \bar{R}_{\mathcal{N}_x}\right), \qquad n_x = \min\{|\mathcal{T}_x|, |\mathcal{N}_x|\}. \tag{7}$$

The factor $n_x/(n_x + \eta)$ shrinks the reward contrast according to the smaller group size, limiting the correction when either reward estimate is based on few trajectories. The parameter $\eta \geq 0$ controls the shrinkage strength; $U_x = 0$ when either group is empty. We apply this contrast to tokens that specify the response action, concentrating the correction at the tool-calling choice. Within each prompt group, let $\mathcal{Z}_{\text{dec}}$ contain the tokens forming tool-call opening markers (e.g., `<python_code>`) in $\mathcal{T}_x$, and $\mathcal{Z}_{\text{dir}}$ contain the tokens forming answer-opening markers (e.g., `<answer>`) in $\mathcal{N}_x$. Decision Contrast assigns corrections $+U_x$ and $-U_x$ to token advantages in $\mathcal{Z}_{\text{dec}}$ and $\mathcal{Z}_{\text{dir}}$, respectively.

This correction changes the difference between tool and direct-answer decision tokens' mean advantages by $2U_x$. Higher sampled rewards for tool use therefore increase the advantage assigned to tool-decision tokens relative to direct-answer decision tokens; higher rewards for direct answering reverse the correction. The comparison thus provides a prompt-specific preference between the two response modes, with each trajectory contributing equally to its mode's reward estimate.

4.3 Supervised Replay

To retain tool-call generation during training, Supervised Replay reuses previously sampled calls as direct supervision targets. At each policy update, we add context-response pairs from tool-calling

turns in trajectories with correct final answers to a fixed-capacity replay buffer, discarding the oldest entries when the capacity is exceeded. Each stored pair (c, y) consists of the preceding interaction context c, including any tool observations, and the model-generated response y containing a tool call.

We uniformly sample a replay minibatch $\mathcal{D}_{\text{rep}}$ from this buffer. For each response y, a binary mask $m_t^{\text{rep}}(c, y)$ identifies the tool-call tokens used as prediction targets. To train the current policy π_θ to reproduce these calls under their original contexts, we minimize their average negative log-likelihood:

$$\mathcal{L}_{\text{rep}}(\theta) = -\frac{1}{M_{\text{rep}}} \sum_{(c,y)\in\mathcal{D}_{\text{rep}}} \sum_{t=1}^{|y|} m_t^{\text{rep}}(c, y) \log \pi_\theta(y_t \mid c, y_{<t}), \tag{8}$$

where M_{rep} counts the tool-call tokens in the minibatch, and y_t and $y_{<t}$ denote the current token and preceding response tokens, respectively. Preceding reasoning remains in the conditioning context.

Minimizing this loss reinforces the recorded tool-call continuations under their original contexts. Replay retains these targets across policy updates, so successful trajectories continue to supervise tool-call generation as the rollout distribution evolves, complementing the advantage adjustments.

4.4 Training Objective

Transition-aware advantages. We add the projection offset to tool-call token advantages and the signed Decision Contrast signal to decision-token advantages, applying both corrections at tool-call openings. For token (i, h, t) in trajectory i sampled for prompt x, the original advantage A_i becomes

$$A_{i,h,t}^{\text{TACA}} = A_i + \underbrace{b_{x,\mathcal{Z}}\, \mathbf{1}\{(i,h,t) \in \mathcal{Z}\}}_{\text{Advantage Projection}} + \underbrace{\lambda_{\text{dc}} U_x \left[\mathbf{1}\{(i,h,t) \in \mathcal{Z}_{\text{dec}}\} - \mathbf{1}\{(i,h,t) \in \mathcal{Z}_{\text{dir}}\}\right]}_{\text{Decision Contrast}}, \tag{9}$$

where $\mathbf{1}\{\cdot\}$ denotes an indicator and $\lambda_{\text{dc}} \geq 0$ controls the strength of Decision Contrast.

Policy optimization. The TACA advantages enter the token-mean policy objective, together with the auxiliary replay loss. For a rollout batch, let S_i denote the number of generated turns in trajectory i, $\ell_{i,h}$ the number of model-generated tokens in turn h, and $N = \sum_i \sum_{h=1}^{S_i} \ell_{i,h}$ their total count. With $\lambda_{\text{rep}} \geq 0$ weighting replay supervision, the policy maximizes

$$\mathcal{J}_{\text{TACA}}(\pi_\theta) = \frac{1}{N} \sum_i \sum_{h=1}^{S_i} \sum_{t=1}^{\ell_{i,h}} \left[\phi\big(\rho_{i,h,t}, A_{i,h,t}^{\text{TACA}}\big) - \beta\, \mathbb{D}_{\text{KL}}[\pi_\theta \| \pi_{\text{ref}}]_{i,h,t}\right] - \lambda_{\text{rep}} \mathcal{L}_{\text{rep}}(\theta). \tag{10}$$

Within each prompt group, TACA preserves reward-dependent advantage differences between tokens receiving the same adjustment, while allowing reasoning and tool interaction to receive different supervision. An unsuccessful trajectory can receive an upward adjustment at tool initiation when the reward contrast favors tool use, while tool calls from correct trajectories provide replay supervision across updates. The transition from reasoning to tool interaction is trained through response comparisons and replay from correct trajectories, using evidence beyond one rollout's outcome.

5 Experiments

5.1 Experimental Setup

Training setup. We post-train Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct [38], and Llama-3.2-1B-Instruct on MATH [13]. We use AlphaApollo [61] for multi-turn agentic training, with code execution as the primary reasoning tool. Training uses four A800 GPUs, 32 prompts per batch, eight rollouts per prompt, and a learning rate of 10^{-6}. Trajectories contain at most six generated turns, with up to 2,048 response tokens per turn and a three-second execution timeout. We also train Qwen2.5-1.5B-Instruct separately on TACO [20] and WebShop [52] to evaluate coding and interactive decision-making, respectively. All tasks use terminal rewards.

Evaluation setup. We evaluate on MATH500 [13], AIME 2024, AIME 2025, AMC 2023, OlympiadBench [12], and GSM8K [2]. We sample 32 responses per question at temperature 0.6 with top-p sampling and report benchmark accuracies and their unweighted mean. TACO reports unit-test scores on its test set, while WebShop reports success rate and task score on its official split.

Baselines. We compare TACA with non-agentic GRPO [34] and agentic methods ReTool [5], MT-GRPO [6], GiGPO [6], and SimpleTIR [51]. Base-model inference and agentic prompting serve as training-free references. Training and evaluation settings are matched for each model-task pair. Further experimental details for TACA and the baseline methods are provided in Appendix I.

Table 1: Main results of post-training experiments (accuracy, in %). Bold and underlined values indicate the best and second-best results within each model, respectively.

Methods	Benchmarks						Average
	AIME24	AIME25	MATH500	AMC23	OlympiadBench	GSM8K	
Qwen2.5-1.5B-Instruct							
Training-free							
Base model	1.09	0.35	38.83	15.56	10.52	59.42	20.96
Agentic prompting	0.20	0.0	10.46	3.44	2.69	16.29	5.51
Non-agentic training							
GRPO	3.03	0.69	50.10	26.02	17.60	<u>72.55</u>	28.33
Agentic training							
ReTool	<u>3.81</u>	1.08	<u>51.90</u>	27.34	16.65	70.74	28.59
Turn-level GRPO	3.26	0.97	50.70	**27.50**	<u>17.88</u>	71.49	<u>28.63</u>
SimpleTIR	3.04	<u>1.38</u>	49.54	22.99	17.39	70.48	27.47
TACA (ours)	**8.43**	**8.12**	**59.14**	<u>27.42</u>	**22.20**	**74.15**	**33.24**
Qwen2.5-3B-Instruct							
Training-free							
Base model	3.45	1.04	56.65	29.32	22.15	76.76	31.56
Agentic prompting	4.16	1.13	38.52	17.43	14.48	45.27	20.17
Non-agentic training							
GRPO	5.55	2.29	<u>62.45</u>	41.02	24.99	81.94	36.37
Agentic training							
ReTool	6.04	6.62	61.03	36.59	27.34	82.95	36.76
Turn-level GRPO	<u>13.57</u>	<u>11.21</u>	58.87	<u>47.67</u>	28.51	79.57	<u>39.90</u>
SimpleTIR	10.76	9.51	61.37	41.16	<u>31.67</u>	**83.52**	39.67
TACA (ours)	**15.56**	**15.00**	**64.51**	**47.96**	**34.62**	<u>83.42</u>	**43.51**

Table 2: Coding and interactive task performance on Qwen2.5-1.5B-Instruct. TACO reports unit-test scores by source and their unweighted average. WebShop reports success rate (SR) and task score.

Method	TACO					WebShop	
	HackerRank	HackerEarth	CodeForces	CodeChef	Avg.	SR (%)	Score
MT-GRPO	59.80	25.00	48.13	34.98	41.98	56.80	75.80
GiGPO	–	–	–	–	–	65.00	83.10
TACA	**64.71**	**50.00**	**58.98**	**41.12**	**53.70**	**67.20**	**85.80**

5.2 Main Results

TACA improves the effectiveness of agentic training. Table 1 shows that TACA achieves the highest average accuracy on both Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct, reaching 33.24% and 43.51%, with the best score on ten of twelve model–benchmark pairs. Agentic prompting alone lowers average accuracy relative to the base models, whereas TACA exceeds non-agentic GRPO by 4.91 and 7.14 percentage points, respectively. Among the agentic training methods, TACA also improves over MT-GRPO by 4.61 and 3.61 percentage points. *These gains over MT-GRPO suggest that TACA makes terminal-reward supervision more effective for learning tool-assisted reasoning.*

Reasoning gains extend to challenging competition problems. On Qwen2.5-3B-Instruct, TACA exceeds the strongest competing result on AIME 2024, AIME 2025, and OlympiadBench by 1.99, 3.79, and 2.95 percentage points, respectively (Table 1). The relative gains are particularly pronounced for the 1.5B model, which improves on the strongest baseline from 3.81% to 8.43% on AIME 2024 and from 1.38% to 8.12% on AIME 2025. The resulting 1.5B TACA policy also outperforms the larger 3B ReTool model on both AIME benchmarks. *This comparison suggests that better agentic training can help smaller models close the gap with larger models on difficult reasoning tasks.*

TACA improves functional correctness across coding problem sources. Table 2 compares coding policies on the TACO test set. TACA raises the mean unit-test score from 41.98 to 53.70, improving all four problem sources: HackerRank, HackerEarth, CodeForces, and CodeChef. The gains range from 4.91 points on HackerRank to 25.00 on HackerEarth. *TACA's training objective remains effective when the training signal measures the functional correctness of generated programs.*

TACA improves successful completion in interactive decision-making. On WebShop, TACA increases success rate from 56.8% to 67.2% and task score from 75.8 to 85.8 relative to MT-GRPO

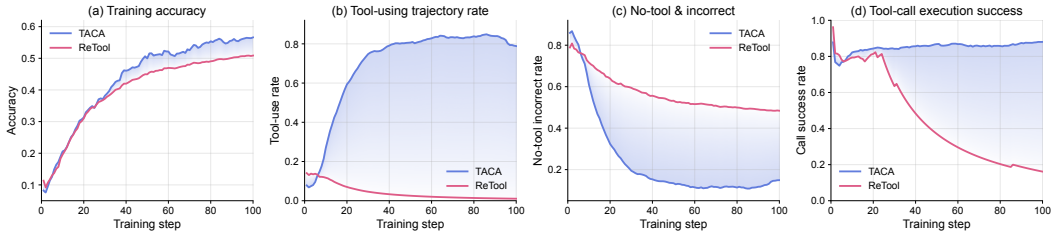

Figure 7: Training dynamics on Qwen2.5-1.5B-Instruct. (a) Both methods improve training accuracy. TACA maintains (b) substantial tool use, (c) a smaller share of no-tool failures, and (d) a higher logged execution-success metric. Appendix I.5 defines the metrics and smoothing.

Table 3: Component ablations on Qwen2.5-1.5B-Instruct. Scores are reported in accuracy (%).

Variant	MATH500	AIME24	AIME25	AMC23	OlympiadBench	GSM8K	Avg.	ΔAvg.
TACA (full)	**59.14**	**8.43**	**8.12**	**27.42**	**22.20**	**74.15**	**33.24**	–
w/o Advantage Projection	53.74	6.55	3.49	25.98	19.16	72.71	30.27	−2.97
w/o Decision Contrast	53.35	6.74	5.24	26.58	18.89	72.67	30.58	−2.66
w/o Supervised Replay	52.68	3.58	3.36	27.31	20.95	73.37	30.21	−3.03

(Table 2). It also exceeds GiGPO by 2.2 percentage points in success rate and 2.7 points in task score. The joint improvement links higher task rewards to more completed goals in an environment where actions determine subsequent observations. *The benefit of transition-aware training thus extends to sequential decisions grounded in external environment feedback.* Together with TACO, this result establishes TACA's applicability across distinct action spaces and terminal reward functions.

5.3 Training Dynamics

TACA improves training accuracy while retaining tool use. Figure 7a,b compares learning progress and interaction behavior on Qwen2.5-1.5B-Instruct. Both methods improve training accuracy, but their tool-use rates evolve in opposite directions: TACA maintains substantial tool use, while ReTool increasingly answers without tools. This divergence changes which actions and environment responses enter subsequent training batches. *Improving terminal rewards can accompany the loss of tool interaction, making rollout composition essential to evaluating agentic learning.* TACA sustains an interactive policy while attaining the stronger held-out performance in Table 1.

Tool-using trajectories account for most successful solutions. Figure 7c shows that TACA samples fewer trajectories that both omit tools and answer incorrectly. The joint rollout statistics further connect this shift to successful reasoning: over the displayed training window, tool-using correct trajectories constitute 42.00% of all TACA trajectories and 87.34% of its correct trajectories (Appendix I.5). The sustained tool-using mode therefore provides a substantial population of correct interaction trajectories in the training batches. *Tool interaction remains embedded in the model's successful solution strategies as training progresses.* This connects the maintained action support in Figure 7b to the task outcomes in panels (a) and (c).

Sustained interaction supplies executable tool calls. Figure 7d tracks the logged execution-success metric alongside the tool-use dynamics. Pooling actual calls over the same training window, TACA executes 47,391 of 54,924 calls successfully, an execution-success rate of 86.28%, compared with 79.45% for ReTool (Appendix I.5). The sampled interactions therefore provide a continuing source of executable actions and environment responses. *TACA maintains both the availability of tool interaction and the ability to carry it out, preserving opportunities to learn from external feedback.*

5.4 Ablation Studies

Both advantage adjustments contribute beyond replay supervision. Table 3 removes one component at a time from TACA on Qwen2.5-1.5B. Removing Advantage Projection lowers average accuracy from 33.24% to 30.27%; removing Decision Contrast lowers it to 30.58%. Both removals reduce accuracy on every benchmark, including MATH500 drops of 5.40 and 5.79 percentage points. Since Supervised Replay remains active in both variants, these comparisons isolate gains from each advantage adjustment in the presence of replay. *Projecting tool-call advantages and guiding tool-use decisions through reward comparisons each improve performance alongside replay supervision.* This supports treating tool-call credit and response-mode selection as complementary parts of optimization.

Supervised Replay adds gains to both advantage adjustments. Removing Supervised Replay reduces mean accuracy by 3.03 percentage points and lowers all six benchmark scores (Table 3). The

Table 4: Mathematical reasoning accuracy (%) across model size and family. Avg. is the six-benchmark mean; bold marks the best result within each model.

Method	MATH500	AIME24	AIME25	AMC23	OlympiadBench	GSM8K	Avg.
Qwen2.5-7B-Instruct							
MT-GRPO	68.92	11.86	8.02	44.11	33.77	**88.84**	42.59
TACA	**70.77**	**18.35**	**13.43**	**47.78**	**36.97**	88.51	**45.97**
Llama-3.2-1B-Instruct							
MT-GRPO	24.04	0.79	0.00	10.12	5.14	41.58	13.61
TACA	**28.07**	**1.74**	**0.31**	**10.93**	**6.42**	**43.54**	**15.17**

largest losses occur on MATH500, AIME24, and AIME25, at 6.46, 4.85, and 4.76 points, respectively. With both advantage adjustments enabled, replay adds supervision on stored calls across policy updates. *Replay supervision complements reward-based credit assignment by retaining a learning signal for tool-call generation across changing rollout distributions.* The full method combines this persistent supervision with the prompt-specific adjustments supplied by the current batch.

The gains persist across model size and family. Table 4 extends the MT-GRPO comparison to Qwen2.5-7B-Instruct and Llama-3.2-1B-Instruct. TACA raises average accuracy from 42.59% to 45.97% on the 7B model and from 13.61% to 15.17% on Llama, improving five of six and all six benchmarks, respectively. These results complement the 1.5B and 3B comparisons: average gains hold for all four evaluated models. *Transition-aware credit assignment remains useful both as model capacity increases and when the underlying model family changes.*

6 Related Work

Agentic learning for LLMs. Agentic LLMs combine reasoning with tool interaction and iterative feedback, using observations and critiques to guide subsequent generation [53, 35, 27, 8]. Tool-use behavior can be learned through self-supervision, demonstrations, and human feedback [31, 28], while policy-gradient methods optimize reasoning policies using task rewards [32, 34, 54]. Supervision can assess complete solutions through learned verifiers [2] or intermediate reasoning steps through process feedback [39, 24]. Recent work extends reinforcement learning to multi-turn tool use and interactive environments, where generated actions determine the observations available for subsequent reasoning [51, 18, 44, 5, 58, 33, 36, 57]. Within these systems, reward design and credit assignment govern how task feedback supervises the sequence of actions leading to an outcome [29, 6, 42, 55].

Training degradation and collapse in agentic RL. Training degradation in agentic RL affects both interaction behavior and policy optimization. Reported patterns include empty turns and gradient spikes [51, 42], repetitive reasoning and reduced exploration [44, 55], unstable entropy dynamics [50, 4], and declining response likelihoods [3]. Stabilization methods adapt the granularity of policy updates and control off-policy variance [19, 9], while local credit-assignment methods refine the learning signals received by individual interaction steps [6, 46, 40]. Related studies examine how exploration and multi-task training influence the trajectories available for learning [57, 18]. Our analysis connects rollout structure to the aggregate terminal-reward credit assigned to tool-call tokens. We characterize when this credit becomes negative and how trajectory weighting and failure-driven retries affect its allocation during multi-turn optimization. Appendix E provides a detailed discussion.

7 Conclusion

We analyzed training degradation in multi-turn agentic reinforcement learning through the credit assigned to tool interactions. In MT-GRPO, incorrect trajectories containing more tool tokens per turn than correct trajectories can produce negative aggregate tool credit. Token-mean aggregation can amplify this imbalance through length weighting, while failure-driven retries extend terminal supervision to subsequent recovery attempts, including successful tool executions. Motivated by these findings, we proposed Transition-Aware Credit Assignment (TACA), which adjusts advantages for tool-call generation and tool-use decisions and sustains supervision through replay. Experiments across model sizes, families, and tasks demonstrate improved performance and sustained tool interaction, with ablations supporting the complementary contributions of advantage adjustments and replay supervision. These findings suggest that the effectiveness of terminal-reward training depends on how credit is allocated to the intermediate transitions between reasoning and tool interaction.

Acknowledgements

XYL, ZKZ, JZN, CTC, and BH were supported by NSFC Major Research Plan No. 92570109, RGC Research Fellow Scheme No. RFS2627-2S02, RGC Young Collaborative Research Grant No. C2005-24Y, RGC General Research Fund No. 12202026, RGC General Research Fund No. 12200725, NSFC General Program No. 62376235, Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme, HKBU Faculty Niche Research Areas No. RC-FNRA-IG/25-26/SCI/05, and HKBU CSD Departmental Incentive Scheme. JCY is supported by National Natural Science Foundation of China (No. 62676245), the Lingang Laboratory (Grant No. LGL-8889-01), STCSM (No. 22DZ2229005) and 111 plan (No. BP0719010).

References

[1] Chentao Cao, Xiaojun Xu, Bo Han, and Hang Li. Reasoned safety alignment: Ensuring jailbreak defense via answer-then-check. In *ICLR*, 2026.

[2] 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.

[3] Wenlong Deng, Yushu Li, Boying Gong, Yi Ren, Christos Thrampoulidis, and Xiaoxiao Li. On group relative policy optimization collapse in agent search: The lazy likelihood-displacement. *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, 2026. URL https://arxiv.org/abs/2512.04220.

[4] 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, 2025. URL https://arxiv.org/abs/2510.14545.

[5] 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, 2025. URL https://arxiv.org/abs/2504.11536.

[6] 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.

[7] 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*, 2026.

[8] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

[9] Chenyang Gu, Yewen Pu, Bruce Yang, Xiaofan Li, and Huan Gao. Dspo: Stable and efficient policy optimization for agentic search and reasoning, 2026. URL https://arxiv.org/abs/2510.09255.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang,

Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

[11] Yupu Hao, Zhuoran Jin, Huanxuan Liao, Kang Liu, and Jun Zhao. Why multi-step tool-use reinforcement learning collapses and how supervisory signals fix it, 2026. URL https://arxiv.org/abs/2606.26027.

[12] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

[13] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

[14] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. URL https://arxiv.org/abs/2503.24290.

[15] Tangyu Jiang, Haodi Wang, and Rongfang Bie. Meco: Zero-shot nas with one data and single forward pass via minimum eigenvalue of correlation. *Advances in neural information processing systems*, 36:61020–61047, 2023.

[16] Tangyu Jiang, Haodi Wang, Rongfang Bie, and Chun Yuan. Accelerating zero-shot nas with feature map-based proxy and operation scoring function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(11):9876–9893, 2025.

[17] Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbić. Meta-RL induces exploration in language agents, 2025. URL https://arxiv.org/abs/2512.16848.

[18] 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.

[19] Chenliang Li, Adel Elmahdy, Alex Boyd, Zhongruo Wang, Siliang Zeng, Alfredo Garcia, Parminder Bhatia, Taha Kass-Hout, Cao Xiao, and Mingyi Hong. Stabilizing off-policy training for long-horizon llm agent via turn-level importance sampling and clipping-triggered normalization, 2026. URL https://arxiv.org/abs/2511.20718.

[20] Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset, 2023. URL https://arxiv.org/abs/2312.14852.

[21] Xuan Li, Zhanke Zhou, Zongze Li, Jiangchao Yao, Tongliang Liu, and Bo Han. Learning to evolve: Open-ended molecular optimization with progress-shaped RL. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026. URL https://openreview.net/forum?id=qEe4ypDaQp.

[22] Xuan Li, Zhanke Zhou, Zongze Li, Jiangchao Yao, Yu Rong, Lu Zhang, and Bo Han. Reference-guided policy optimization for molecular optimization via llm reasoning. In *International Conference on Learning Representations*, 2026.

[23] Xuefeng Li, Haoyang Zou, and Pengfei Liu. ToRL: Scaling tool-integrated RL, 2025. URL https://arxiv.org/abs/2503.23383.

[24] 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, pages 39578–39601, 2024.

[25] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

[26] 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.

[27] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems, 2023*, 2023. URL https://arxiv.org/abs/2303.17651.

[28] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

[29] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

[30] Yulei Qin, Xiaoyu Tan, Zhengbao He, Gang Li, Haojia Lin, Zongyi Li, Zihan Xu, Yuchen Shi, Siqi Cai, Renting Rui, Shaofei Cai, Yuzheng Cai, Xuan Zhang, Sheng Ye, Ke Li, and Xing Sun. Learn the ropes, then trust the wins: Self-imitation with progressive exploration for agentic reinforcement learning, 2025. URL https://arxiv.org/abs/2509.22601.

[31] 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.

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

[33] Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, Ying Xin, Ziming Miao, Scarlett Li, Fan Yang, and Mao Yang. rstar2-agent: Agentic reasoning technical report, 2025. URL https://arxiv.org/abs/2508.20722.

[34] 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.

[35] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[36] Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.01441.

[37] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Weixin Xu, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, Zonghan Yang, and Zongyu Lin. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

[38] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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

[40] Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents, 2026. URL https://arxiv.org/abs/2510.14967.

[41] Hongru Wang, Rui Wang, Jushi Kai, Boyang Xue, Yongqi Li, Shijue Huang, Xiaoteng Ma, Jeff Z. Pan, and Amos Storkey. Self-sum: Teaching an agent to decide itself when and what to summarize. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Findings of the Association for Computational Linguistics: ACL 2026*, pages 9178–9192, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.447. URL https://aclanthology.org/2026.findings-acl.447/.

[42] Ruiyi Wang and Prithviraj Ammanabrolu. A practitioner's guide to multi-turn agentic reinforcement learning, 2025. URL https://arxiv.org/abs/2510.01132.

[43] Zhitong Wang, Songze Li, Hao Peng, Shuzheng Si, Yi Wang, Maosong Sun, and Juanzi Li. EnvRL: Learn from environment dynamics in agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2606.17680.

[44] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

[45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[46] Quan Wei, Siliang Zeng, Chenliang Li, 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 turn-level reward design, 2025. URL https://arxiv.org/abs/2505.11821.

[47] Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

[48] Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2509.08755.

[49] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

[50] Wujiang Xu, Wentian Zhao, Zhenting Wang, Yu-Jhe Li, Can Jin, Mingyu Jin, Kai Mei, Kun Wan, and Dimitris N. Metaxas. Epo: Entropy-regularized policy optimization for llm agents reinforcement learning, 2026. URL https://arxiv.org/abs/2509.22576.

[51] Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning, 2025. URL https://arxiv.org/abs/2509.02479.

[52] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents, 2022. URL https://arxiv.org/abs/2207.01206.

[53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

[54] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan 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, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

[55] Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning, 2025. URL https://arxiv.org/abs/2510.11701.

[56] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025. URL https://arxiv.org/abs/2503.18892.

[57] Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework, 2025. URL https://arxiv.org/abs/2510.04206.

[58] Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning, 2025. URL https://arxiv.org/abs/2505.00024.

[59] Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. RLVMR: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents, 2025. URL https://arxiv.org/abs/2507.22844.

[60] Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems*, 37:123846–123910, 2024.

[61] Zhanke Zhou, Chentao Cao, Xiao Feng, Xuan Li, Zongze Li, Xiangyu Lu, Jiangchao Yao, Weikai Huang, Tian Cheng, Jianghangfan Zhang, et al. Alphaapollo: A system for deep agentic reasoning. *arXiv preprint arXiv:2510.06261*, 2025.

[62] Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LCaTpVuvpj.

[63] Zhanke Zhou, Zongze Li, Weikai Huang, Xuan Li, Chentao Cao, Xiao Feng, Xiangyu Lu, Jinbo Hu, Yi Xie, Nico Pelleriti, et al. Alphadiana: A measurement system for agents in verifiable reasoning tasks. *arXiv preprint arXiv:2610.20675*, 2026.

[64] Zhanke Zhou, Xiangyu Lu, Chentao Cao, Brando Miranda, Tongliang Liu, Bo Han, and Sanmi Koyejo. The easy, the hard, and the learnable: Confidence and difficulty-adaptive policy optimization for llm reasoning. *arXiv preprint arXiv:2606.07950*, 2026.

[65] Zhanke Zhou, Zhaocheng Zhu, Xuan Li, Mikhail Galkin, Xiao Feng, Sanmi Koyejo, Jian Tang, and Bo Han. Landscape of thoughts: Visualizing the reasoning process of large language models. In *ICLR*, 2026.

Appendix

A Ethics Statement **19**

B LLM Usage Disclosure **19**

C Impact Statement **19**

D Limitations **19**

E Related Work **19**

F Extended Analysis **20**

F.1 Terminal-Reward Coefficients in the Policy Gradient 21

F.2 Proof of the Tool-Credit Condition 21

F.3 Credit Aggregation across Prompt Groups 22

F.4 Length-Dependent Reward Normalization 22

F.5 Length Weighting and the Aggregation-Induced Credit Shift 23

F.6 Reward Normalization and Loss Reduction 24

F.7 Terminal-Reward Credit on Recovery Attempts 24

G Additional Analysis Details **25**

G.1 Empirical Setup and Analysis Window 25

G.2 Overview Metrics and Display Conventions 25

G.3 Tool-Token Density and Aggregate Credit 25

G.4 Reward Means and Paired Credit Aggregation 26

G.5 Execution-Conditioned Continuation 27

G.6 Retry Statistics by Final Correctness 27

H TACA Method Details **27**

H.1 Tool-Call Tokens and Decision Positions 28

H.2 Advantage Projection as a Minimal Adjustment 28

H.3 Shrinkage and Credit in Decision Contrast 29

H.4 Supervised Replay in the Original Interaction Context 29

H.5 Training Procedure 30

I Implementation Details **30**

I.1 Training Configuration 30

I.2 Tasks, Rewards, and Evaluation 31

I.3 Interaction Prompts 32
I.4 Normalization Comparison 33
I.5 Training Diagnostics 33

J NeurIPS Paper Checklist **34**

A Ethics Statement

This study examines reinforcement learning for tool-using language models using existing benchmarks. The experiments use automated task evaluation and involve no human participants.

B LLM Usage Disclosure

This manuscript was prepared with the assistance of LLMs, which were used for refining content and ensuring grammatical accuracy. The authors take full responsibility for the entire content of the manuscript.

C Impact Statement

This work contributes a diagnostic account and mitigation of training collapse in multi-turn agentic reinforcement learning with tool use. By identifying transition-blind credit assignment as a structural cause and proposing Transition-Aware Credit Assignment (TACA), we aim to make agentic post-training of language models more reliable and reproducible, reducing wasted compute on silently collapsing runs and lowering the barrier for smaller research groups developing tool-using reasoning systems.

D Limitations

The experiments cover mathematical reasoning, TACO coding, and WebShop with Qwen models up to 7B and Llama-3.2-1B. Larger models, long-horizon software engineering, multi-agent coordination, and variability across training seeds are directions for further evaluation. Reward contrasts depend on sufficient sampled support, and Supervised Replay depends on the quality of recorded tool calls. Studying convergence of the full regularized objective is a direction for theoretical development.

E Related Work

Reasoning and outcome supervision. Chain-of-thought prompting elicits intermediate reasoning steps [45]. Errors in demonstration rationales can reduce answer accuracy [60], and visualizations of sampled reasoning paths reveal where correct and incorrect solutions diverge [65]. At the level of complete solutions, learned verifiers rank candidates by predicted correctness [2]. PPO optimizes a clipped policy objective [32]. DeepSeekMath uses group-relative advantages [34], and DAPO aggregates the loss over generated tokens [54]. In the multi-turn GRPO setting studied here, each tool call inherits the advantage computed from the trajectory's final outcome.

Tool interaction and iterative feedback. Interactive reasoning allows a model to obtain missing information by asking a question [62] or acting on an external environment [53]. Models learn to browse from demonstrations and human feedback [28] and to insert API calls through self-supervision [31]. Models can revise a response using their own critiques or tool feedback [35, 27, 8], or check a proposed answer for safety before deciding whether to return it [1]. In tool use, an execution error becomes part of the context for the next decision. Section 3.4 examines how retries change the remaining trajectory and the actions that receive terminal-reward supervision.

Multi-turn tool-integrated reinforcement learning. Outcome rewards can train a policy to interleave reasoning with code execution [23, 5, 51, 36] or search [18]. Work on tool learning studies how to reward tool use and train the reasoning that precedes a call [29, 58]. Training systems support these interactions through scalable execution and rollout resampling [33] or through multi-environment and asynchronous multi-task training [48, 57]. Studies of these policies examine how environment complexity, reward sparsity, and exploration affect training [42, 55]. Measuring tool use and resource consumption alongside task outcomes helps distinguish agents that reach similar answers through different interactions [63].

Reward normalization and loss aggregation. Multi-turn implementations differ in the samples used to compute reward statistics. The released implementations of ToRL, SimpleTIR, Search-R1, and AgentGym-RL normalize rewards over complete trajectories and use token-mean loss aggregation [23, 51, 18, 48]. LightningRL in Agent Lightning [26] and the GRPO baseline released with GiGPO [6] instead replicate each episode return across its turns before computing group statistics; their released GRPO implementations also use token-mean aggregation. This turn-expanded normalization is retained by the released GRPO baselines of RLVMR [59], SPEAR [30], and LaMer [17], and by the RL component of EnvRL-GRPO [43]. Dr. GRPO removes response-length and reward-standard-deviation normalization to address optimization biases in GRPO [25]. CoDaPO weights questions using rollout confidence and estimated difficulty [64]. Section 2 formalizes the turn-level normalization and token-mean objective studied here. Our analysis separates the resulting turn-weighted reward statistics from the token-proportional trajectory weights introduced by loss aggregation, and characterizes their implications for aggregate credit on tool-call spans.

Process supervision and local credit assignment. Process supervision evaluates intermediate reasoning steps, including through reward models trained on human step-level feedback [39, 24]. In molecular optimization, supervision can take the form of a reference molecule for a proposed solution [22] or a change in molecular properties across successive edits [21]. The granularity of evaluation also matters in neural architecture search, where correlations between feature maps can be used to score candidate networks and their constituent operations before training [15, 16]. In agentic RL, local credit can be inferred from sampled interactions by grouping actions taken in the same state [6] or propagating outcome information through a graph of visited states [7]. Other approaches combine episode-level and local rewards [59, 41], vary the timing of feedback [46], or reward a turn according to the change in the probability of the correct answer [40]. TACA uses terminal rewards to adjust advantages on tool-call and decision tokens and draws on calls from correct trajectories for replay supervision.

Training degradation and stabilization. In multi-turn RL, training degradation can manifest in sampled interactions. RAGEN reports an "echo trap" in which reasoning becomes repetitive and trajectory diversity declines [44]. SimpleTIR links low-probability generations to gradient spikes and filters trajectories containing turns with neither complete code nor a final answer [51]. EPO addresses early low-entropy convergence and later instability in exploration through adaptive entropy regularization [50]; AEPO balances entropy in rollout allocation and policy updates [4]. DSPO combines sequence-level optimization with dynamic sample filtering [9], while SORL uses turn-level importance sampling and clipping-triggered normalization to stabilize off-policy updates [19]. Deng et al. [3] identify stagnating or declining likelihoods of both correct and incorrect responses and introduce likelihood-preserving regularization. Hao et al. [11] associate failures of tool-invocation structure with probability spikes in control tokens and study supervisory signals for stabilizing training.

Our analysis concerns the aggregate terminal-reward credit assigned to tool-call tokens. For mixed-outcome groups with binary rewards, the fully on-policy objective in Section 2 assigns negative aggregate tool credit when incorrect trajectories contain more tool tokens per generated turn than correct trajectories. We further characterize the credit shift induced by token-mean aggregation and examine the additional actions contributed by failure-conditioned retries. These relationships motivate TACA's three components: Advantage Projection, Decision Contrast, and Supervised Replay. The outer objective retains token-proportional trajectory weighting; the corrections act on the credit assigned within that objective.

F Extended Analysis

We derive the credit identities and length-weighting relationships in Section 3. We use the reward statistics and generated-token objective of Section 2. Unless a batch is specified, quantities refer to one prompt group. Advantages are held fixed when differentiating the policy surrogate, and standard deviations use the population convention in Eq. (1).

F.1 Terminal-Reward Coefficients in the Policy Gradient

Let $y_{i,h,t}$ be the generated token at position t in turn h of trajectory i, and let $H_{i,h,t}$ contain its prompt and preceding interaction history. At the rollout policy, the token importance ratio $\rho_{i,h,t}$ equals one and lies inside the clipping interval. For the clipped surrogate ϕ with turn-normalized terminal-reward advantage A_i^{turn}, the derivative is

$$\nabla_\theta \phi(\rho_{i,h,t}, A_i^{\text{turn}})\big|_{\theta=\theta_{\text{old}}} = A_i^{\text{turn}}\, \nabla_\theta \log \pi_\theta(y_{i,h,t} \mid H_{i,h,t})|_{\theta=\theta_{\text{old}}}\,. \tag{11}$$

For a subset $\mathcal{Z}$ of policy-scored generated tokens, its contribution to the on-policy reward-surrogate gradient, normalized by the group's total policy-scored token count N, is

$$g_{\mathcal{Z}} = \frac{1}{N} \sum_{(i,h,t)\in\mathcal{Z}} A_i^{\text{turn}}\, \nabla_\theta \log \pi_\theta(y_{i,h,t} \mid H_{i,h,t})|_{\theta=\theta_{\text{old}}}\,. \tag{12}$$

This gives the coefficient A_i^{turn}/N used in Proposition 3.1. Within a prompt group, the mean and standard deviation are shared, so equal terminal rewards produce equal advantage coefficients. In a mixed binary-reward group, every tool call in an incorrect trajectory receives a negative coefficient, including calls whose execution succeeds. Execution feedback enters subsequent generation contexts; the completed trajectory's reward determines the coefficient assigned to each generated token.

F.2 Proof of the Tool-Credit Condition

For a subset $\mathcal{Z}$ of policy-scored generated tokens, let $z_{i,h}$ count its tokens in turn h and let $z_i = \sum_h z_{i,h}$. Summing the coefficients in Eq. (12) defines its aggregate credit:

$$C(\mathcal{Z}) \coloneqq \frac{1}{N} \sum_i z_i A_i^{\text{turn}}. \tag{13}$$

With S_i generated turns in trajectory i and $M = \sum_i S_i$ turns in the group, $\bar{\ell} = N/M$ is the mean generated-token count per turn. For a mixed group with binary terminal rewards R_i, define the correct and incorrect trajectory sets $\mathcal{P} = \{i : R_i = 1\}$ and $\mathcal{F} = \{i : R_i = 0\}$, their turn counts $M^+ = \sum_{i\in\mathcal{P}} S_i$ and $M^- = \sum_{i\in\mathcal{F}} S_i$, and their subset-token densities

$$\bar{z}^+_{\text{turn}} = \frac{\sum_{i\in\mathcal{P}} z_i}{M^+}, \qquad \bar{z}^-_{\text{turn}} = \frac{\sum_{i\in\mathcal{F}} z_i}{M^-}.$$

Lemma F.1 (Subset-credit factorization)**.** *For a prompt group containing both binary outcomes, with turn-weighted reward standard deviation* σ_{turn}*,*

$$C(\mathcal{Z}) = \frac{\sqrt{M^+M^-}}{N}(\bar{z}^+_{\text{turn}} - \bar{z}^-_{\text{turn}}) = \frac{\sigma_{\text{turn}}}{\bar{\ell}}(\bar{z}^+_{\text{turn}} - \bar{z}^-_{\text{turn}}). \tag{14}$$

Proof. The positive mass in the turn-replicated reward distribution is $p = M^+/M$. Thus $\sigma^2_{\text{turn}} = p(1-p)$ and the advantages of correct and incorrect trajectories are, respectively,

$$A^+_{\text{turn}} = \sqrt{\frac{M^-}{M^+}}, \qquad A^-_{\text{turn}} = -\sqrt{\frac{M^+}{M^-}}.$$

Substituting into Eq. (13) gives

$$\begin{aligned} NC(\mathcal{Z}) &= M^+\bar{z}^+_{\text{turn}}\sqrt{\frac{M^-}{M^+}} - M^-\bar{z}^-_{\text{turn}}\sqrt{\frac{M^+}{M^-}} \\ &= \sqrt{M^+M^-}(\bar{z}^+_{\text{turn}} - \bar{z}^-_{\text{turn}}). \end{aligned}$$

The second equality follows from $\sigma_{\text{turn}} = \sqrt{M^+M^-}/M$ and $\bar{\ell} = N/M$. □

Taking $\mathcal{Z}$ to contain all tool-call tokens gives $z_i = m_i$, the tool-token count in trajectory i, and proves Proposition 3.1. Since $\sigma_{\text{turn}}/\bar{\ell} > 0$, the density difference determines the sign; reward composition and mean turn length scale its magnitude. The factorization also applies to subsets of tool-call tokens, including their opening markers.

Extension to scalar rewards. Let $\mathbb{E}_{\text{turn}}$ and Cov_{turn} denote empirical expectation and covariance over the group's M generated turns, each with weight $1/M$. For a bounded scalar terminal reward with $\sigma_{\text{turn}} > 0$,

$$\begin{aligned} C(\mathcal{Z}) &= \frac{\mathbb{E}_{\text{turn}}[z_{i,h} A_i^{\text{turn}}]}{\bar{\ell}} = \frac{\text{Cov}_{\text{turn}}(z_{i,h}, A_i^{\text{turn}})}{\bar{\ell}} \\ &= \frac{\text{Cov}_{\text{turn}}(z_{i,h}, R_i)}{\bar{\ell}\,\sigma_{\text{turn}}}. \end{aligned} \tag{15}$$

The second equality uses $\mathbb{E}_{\text{turn}}[A_i^{\text{turn}}] = 0$, and the third substitutes the standardized reward. Aggregate subset credit is negative precisely when subset-token counts per turn have negative covariance with terminal reward. For binary rewards, this covariance is $p(1-p)(\bar{z}^+_{\text{turn}} - \bar{z}^-_{\text{turn}})$, recovering Lemma F.1.

Normalization implementations. For advantages of the form $A_i = (R_i - \mu_{\text{turn}})/s$ with a shared positive denominator s, Eq. (15) has denominator $\bar{\ell}s$, and the binary multiplier in Eq. (14) is $\sigma^2_{\text{turn}}/(\bar{\ell}s)$. Sample-standard-deviation normalization and additive numerical stabilization therefore preserve the sign condition. Groups with identical rewards use zero advantages and contribute zero credit.

F.3 Credit Aggregation across Prompt Groups

For prompt x, let $\mathcal{G}_x$ denote the set of sampled trajectory indices. Subscript x identifies quantities computed within this group, including its generated-token count N_x, subset credit $C_x(\mathcal{Z})$, and turn-level advantages $A^{\text{turn}}_{x,i}$. A batch $\mathcal{B}$ has $N_{\mathcal{B}} = \sum_x N_x$ generated tokens. Token-mean aggregation gives

$$C_{\mathcal{B}}(\mathcal{Z}) = \frac{1}{N_{\mathcal{B}}} \sum_x \sum_{i \in \mathcal{G}_x} z_i A^{\text{turn}}_{x,i} = \sum_x \frac{N_x}{N_{\mathcal{B}}} C_x(\mathcal{Z}). \tag{16}$$

For binary rewards, let $\mathcal{X}_{\text{mix}}$ contain the prompt groups with both outcomes. Applying Lemma F.1 within each group yields

$$C_{\mathcal{B}}(\mathcal{Z}) = \frac{1}{N_{\mathcal{B}}} \sum_{x \in \mathcal{X}_{\text{mix}}} M_x \sigma_{x,\text{turn}} \left(\bar{z}^+_{x,\text{turn}} - \bar{z}^-_{x,\text{turn}}\right). \tag{17}$$

The batch sign is determined by the weighted sum of prompt-level density gaps, with positive weights $M_x \sigma_{x,\text{turn}}$. Figure 3(a,b) summarizes pooled tool-token densities, while Figure 3(c) evaluates batch credit using the advantages assigned within each prompt group.

F.4 Length-Dependent Reward Normalization

For a prompt group of G trajectories with binary terminal rewards $R_i \in \{0, 1\}$ and both outcomes present, let K^+ and K^- count correct and incorrect trajectories. Trajectory i contains S_i generated turns; their total is $M = \sum_i S_i$, with M^+ and M^- contributed by correct and incorrect trajectories. The corresponding mean turn counts are $\bar{S}^{\pm} = M^{\pm}/K^{\pm}$. Writing $q = K^+/G$ for the correct-trajectory fraction and $\Delta = \bar{S}^-/\bar{S}^+$ for the ratio of mean turn counts gives

$$\mu_{\text{turn}} = \frac{1}{M} \sum_i S_i R_i = \frac{K^+ \bar{S}^+}{K^+ \bar{S}^+ + K^- \bar{S}^-} = \frac{q}{q + (1-q)\Delta}. \tag{18}$$

The trajectory-weighted reward mean is $\mu_{\text{traj}} = q$, so

$$\mu_{\text{traj}} - \mu_{\text{turn}} = \frac{q(1-q)(\Delta - 1)}{q + (1-q)\Delta}. \tag{19}$$

When $\Delta > 1$, incorrect trajectories constitute $1 - q$ of sampled trajectories but contribute the larger share $1 - \mu_{\text{turn}}$ of generated turns. The difference between these shares is positive and increases with Δ at fixed q. Applying this counting identity to all trajectories in a batch gives the pooled reward means in Figure 8; within each prompt group, the turn-weighted mean enters advantage normalization through Eq. (1).

Lemma F.2 (Length-dependent advantage rescaling)**.** *For the same mixed binary-reward group, let $A^{\pm}_{\text{turn}}$ and $A^{\pm}_{\text{traj}}$ denote the advantages of correct and incorrect trajectories under turn-level and trajectory-level normalization. Then*

$$\frac{A^{+}_{\text{turn}}}{A^{+}_{\text{traj}}} = \sqrt{\Delta}, \qquad \frac{|A^{-}_{\text{turn}}|}{|A^{-}_{\text{traj}}|} = \Delta^{-1/2}. \tag{20}$$

Proof. For binary rewards with positive mass p, the standardized advantages are $\sqrt{(1-p)/p}$ and $-\sqrt{p/(1-p)}$. Trajectory-level normalization uses $p = q$, while turn-level normalization uses $p = \mu_{\text{turn}}$. Substituting

$$\frac{1-\mu_{\text{turn}}}{\mu_{\text{turn}}} = \Delta \frac{1-q}{q}$$

gives both ratios. □

For $\Delta > 1$, turn-level normalization increases positive advantages and reduces negative-advantage magnitudes relative to trajectory-level normalization. The aggregation analysis below uses the resulting A_i^{turn} in both objectives, isolating the contribution of token weighting.

F.5 Length Weighting and the Aggregation-Induced Credit Shift

For G sampled trajectories, let S_i count the generated turns in trajectory i and let $\ell_{i,h}$ count the model-generated tokens included in the objective from turn h. Their trajectory and total token counts are $L_i = \sum_{h=1}^{S_i} \ell_{i,h} > 0$ and $N = \sum_{i=1}^{G} L_i$, respectively. Let $\mathcal{J}_i(\theta)$ be the per-token average of the surrogate and KL terms in trajectory i. Regrouping Eq. (2) gives

$$\mathcal{J}_{\text{MT-GRPO}}(\theta) = \sum_{i=1}^{G} \frac{L_i}{N} \mathcal{J}_i(\theta), \qquad \mathcal{J}_i(\theta) = \frac{1}{L_i} \sum_{h=1}^{S_i} \sum_{t=1}^{\ell_{i,h}} \left[\phi(\rho_{i,h,t}, A_i^{\text{turn}}) - \beta \mathbb{D}_{\text{KL},i,h,t} \right], \tag{21}$$

Here ϕ is the clipped surrogate, $\rho_{i,h,t}$ is the token importance ratio, and β weights the token-level KL term $\mathbb{D}_{\text{KL},i,h,t}$ from Eq. (2). Each trajectory receives weight L_i/N. Applying this weight to its tool credit per generated token, $(m_i/L_i)A_i^{\text{turn}}$, gives the token-mean expression in Eq. (4), where m_i counts its tool-call tokens. Replacing L_i/N by $1/G$ gives the trajectory-mean expression.

Weight assigned to incorrect trajectories. For a prompt group containing both binary outcomes, let $L^{\pm}$ be the total generated-token count in the correct and incorrect slices, which contain $K^{\pm}$ trajectories. Their mean token lengths are $\bar{L}^{\pm} = L^{\pm}/K^{\pm}$, and the group mean is $\bar{L} = N/G = q\bar{L}^{+} + (1-q)\bar{L}^{-}$, where q is the correct-trajectory fraction. The incorrect slice receives weight

$$\frac{L^{-}}{N} = \frac{(1-q)\bar{L}^{-}}{q\bar{L}^{+} + (1-q)\bar{L}^{-}}, \qquad \frac{L^{-}}{N} - (1-q) = \frac{q(1-q)(\bar{L}^{-} - \bar{L}^{+})}{\bar{L}}. \tag{22}$$

Its objective weight exceeds its sampling frequency exactly when incorrect trajectories have greater mean token length. Let $M^{\pm}$ count generated turns in the two slices, with mean tokens per turn $\bar{\ell}^{\pm} = L^{\pm}/M^{\pm}$ and mean turn-count ratio $\Delta = (M^{-}/K^{-})/(M^{+}/K^{+})$. Then $\bar{L}^{-}/\bar{L}^{+} = \Delta\bar{\ell}^{-}/\bar{\ell}^{+}$, separating the effects of turn counts and tokens per turn.

Proof of the credit-shift identity. For a batch $\mathcal{B}$, let z_i count tokens from subset $\mathcal{Z}$ of model-generated tokens in trajectory i. Its subset credit per generated token is $c_i = z_i A_i^{\text{turn}}/L_i$, where A_i^{turn} is computed within its prompt group. Let $\mathbb{E}_{\mathcal{B}}$ and $\text{Cov}_{\mathcal{B}}$ denote the empirical mean and covariance with equal weight $1/|\mathcal{B}|$ per trajectory, and let $\bar{L}_{\mathcal{B}} = \mathbb{E}_{\mathcal{B}}[L_i]$. The two aggregations give

$$\begin{aligned} C^{\text{tok}}_{\mathcal{B}}(\mathcal{Z}) &= \frac{\mathbb{E}_{\mathcal{B}}[L_i c_i]}{\bar{L}_{\mathcal{B}}}, \qquad C^{\text{traj}}_{\mathcal{B}}(\mathcal{Z}) = \mathbb{E}_{\mathcal{B}}[c_i], \\ C^{\text{tok}}_{\mathcal{B}}(\mathcal{Z}) - C^{\text{traj}}_{\mathcal{B}}(\mathcal{Z}) &= \frac{\text{Cov}_{\mathcal{B}}(L_i, c_i)}{\bar{L}_{\mathcal{B}}}. \end{aligned} \tag{23}$$

The token-mean expression follows from $\sum_i L_i c_i / \sum_i L_i = \mathbb{E}_{\mathcal{B}}[L_i c_i]/\bar{L}_{\mathcal{B}}$. Subtracting the trajectory mean gives numerator $\mathbb{E}_{\mathcal{B}}[L_i c_i] - \mathbb{E}_{\mathcal{B}}[L_i]\mathbb{E}_{\mathcal{B}}[c_i]$, which is the empirical covariance. Setting $z_i = m_i$, the tool-call token count, recovers Eq. (4) and proves Eq. (5).

Negative covariance lowers token-mean credit relative to trajectory-mean credit. When $C_{\mathcal{B}}^{\mathrm{traj}}(\mathcal{Z}) > 0$, the exact condition for a reversal to negative credit is

$$\mathrm{Cov}_{\mathcal{B}}(L_i, c_i) < -\bar{L}_{\mathcal{B}} C_{\mathcal{B}}^{\mathrm{traj}}(\mathcal{Z}).$$

When trajectory-mean credit is negative, negative covariance increases its magnitude. Figure 4(a) compares the two aggregations on the same batches and turn-level advantages.

F.6 Reward Normalization and Loss Reduction

Reward normalization determines the trajectory advantage; loss reduction determines the weight of each token's surrogate. The following configurations make these choices explicit:

Reward statistics	Loss reduction	Per-token weight	Total trajectory weight
Trajectory	Trajectory mean	$1/(GL_i)$	$1/G$
Trajectory	Token mean	$1/N$	L_i/N
Turn	Turn mean	$1/(M\ell_{i,h})$	S_i/M
Turn	Token mean	$1/N$	L_i/N

Here turn-mean reduction averages the per-token mean of every generated turn, with $\ell_{i,h} > 0$ for each included turn. For a mixed binary-reward group, $\mathcal{P}$ and $\mathcal{F}$ index correct and incorrect trajectories. The corresponding trajectory and turn counts are $K^{\pm}$ and $M^{\pm}$. Define the average subset counts per trajectory and the average subset-token fractions:

$$\bar{z}^{+}_{\mathrm{traj}} = \frac{1}{K^{+}} \sum_{i \in \mathcal{P}} z_i, \qquad \bar{z}^{-}_{\mathrm{traj}} = \frac{1}{K^{-}} \sum_{i \in \mathcal{F}} z_i,$$

$$\bar{f}^{+}_{\mathrm{traj}} = \frac{1}{K^{+}} \sum_{i \in \mathcal{P}} \frac{z_i}{L_i}, \qquad \bar{f}^{-}_{\mathrm{traj}} = \frac{1}{K^{-}} \sum_{i \in \mathcal{F}} \frac{z_i}{L_i},$$

$$\bar{f}^{+}_{\mathrm{turn}} = \frac{1}{M^{+}} \sum_{i \in \mathcal{P}} \sum_{h=1}^{S_i} \frac{z_{i,h}}{\ell_{i,h}}, \quad \bar{f}^{-}_{\mathrm{turn}} = \frac{1}{M^{-}} \sum_{i \in \mathcal{F}} \sum_{h=1}^{S_i} \frac{z_{i,h}}{\ell_{i,h}}.$$

Let $\sigma_{\mathrm{traj}} = \sqrt{q(1-q)}$, $\bar{L} = N/G$, and $\bar{\ell} = N/M$. In the credit expressions below, the first superscript specifies the reward-normalization unit: traj for trajectories or turn for generated turns. The second specifies the loss reduction: unit gives equal weight to each such unit's token-mean contribution, while tok averages all generated-token contributions. Applying the factorization in Lemma F.1 with the corresponding normalization and reduction gives

$$C^{\mathrm{traj,unit}}(\mathcal{Z}) = \sigma_{\mathrm{traj}}(\bar{f}^{+}_{\mathrm{traj}} - \bar{f}^{-}_{\mathrm{traj}}), \tag{24}$$

$$C^{\mathrm{traj,tok}}(\mathcal{Z}) = \frac{\sigma_{\mathrm{traj}}}{\bar{L}}(\bar{z}^{+}_{\mathrm{traj}} - \bar{z}^{-}_{\mathrm{traj}}), \tag{25}$$

$$C^{\mathrm{turn,unit}}(\mathcal{Z}) = \sigma_{\mathrm{turn}}(\bar{f}^{+}_{\mathrm{turn}} - \bar{f}^{-}_{\mathrm{turn}}), \tag{26}$$

$$C^{\mathrm{turn,tok}}(\mathcal{Z}) = \frac{\sigma_{\mathrm{turn}}}{\bar{\ell}}(\bar{z}^{+}_{\mathrm{turn}} - \bar{z}^{-}_{\mathrm{turn}}). \tag{27}$$

For a unit-mean reduction, the proof uses each trajectory's or turn's subset-token fraction. For token-mean reduction, it uses the subset-token count divided by mean unit length. Consequently, count differences determine the sign under token means, while within-unit fraction differences determine the sign under unit means. Each row uses its specified reward normalization; Eq. (23) compares reductions with the same advantages.

F.7 Terminal-Reward Credit on Recovery Attempts

Section 3.4 connects execution feedback to subsequent tool actions. Consider tool calls in the generated turn immediately following an execution failure, corresponding to the continuation event in Figure 5(a). In completed trajectory i, let m_i count all tool-call tokens and r_i count tokens in these recovery calls. Partitioning its tool tokens into recovery calls and other calls gives

$$C_{\mathrm{tool}} = C_{\mathrm{rec}} + C_{\mathrm{other}}, \qquad C_{\mathrm{rec}} = \frac{1}{N} \sum_i r_i A_i^{\mathrm{turn}}, \qquad C_{\mathrm{other}} = \frac{1}{N} \sum_i (m_i - r_i) A_i^{\mathrm{turn}}. \tag{28}$$

In a mixed binary-reward group, correct and incorrect trajectories contribute M^+ and M^- generated turns. Summing recovery credit over the incorrect trajectories $i \in \mathcal{F}$ gives

$$C^-_{\text{rec}} = \frac{1}{N} \sum_{i \in \mathcal{F}} r_i A_i^{\text{turn}} = -\frac{1}{N} \sqrt{\frac{M^+}{M^-}} \sum_{i \in \mathcal{F}} r_i. \tag{29}$$

It is strictly negative whenever an incorrect trajectory contains recovery tokens. Each token in a recovery call that executes successfully receives this same negative coefficient. Thus a successful recovery action inherits negative credit when the completed trajectory's answer is incorrect.

Recovery attempts also contribute to the generated-turn and generated-token counts used in reward normalization and objective aggregation. Figure 5(a,b) measures subsequent calls and generation length conditional on execution outcomes; panel (c) reports retry rates by final correctness. The credit decomposition specifies how the resulting recovery actions enter terminal-reward supervision.

G Additional Analysis Details

G.1 Empirical Setup and Analysis Window

Figures 2–4 analyze Qwen2.5-3B-Instruct trained on DAPO-Math with MT-GRPO and token-mean aggregation. Each batch contains 16 prompts with eight rollouts per prompt, and the maximum interaction length is eight environment steps. Validation accuracy measures final-answer correctness. The evaluation trajectories cover MATH-500, AIME24, AIME25, AMC23, OlympiadBench, MinervaMath, and GSM8K.

The figures display training steps 1–500, with one training batch per step. Validation accuracy peaks at 38.95% at step 200 and reaches 26.64% at step 500; the non-agentic reference is 35%. The peak defines the two reporting intervals, steps 1–200 and 201–500.

G.2 Overview Metrics and Display Conventions

Trajectory and execution statistics. Trajectory-level metrics count each sampled trajectory once. Correctness is determined by the terminal episode reward, with positive reward indicating a correct trajectory. Tool-use rate is the fraction of trajectories containing at least one tool call; execution success rate is the number of successful executions divided by the number of tool calls. Mean tool-call counts are computed separately for correct and incorrect trajectories. In this environment, each trajectory contains one terminal non-tool turn, so tool-call counts equal generated-turn counts minus one. Over steps 1–200 and 201–500, respectively, the raw mean tool-use rates are 67.77% and 72.10%, and execution success rates are 69.41% and 48.41%. Correct trajectories average 1.134 and 1.158 tool calls, and incorrect trajectories average 1.131 and 2.026 tool calls.

Time-series display and summaries. Validation is plotted at recorded checkpoints. Training curves in Figure 2 use exponential moving average (EMA) smoothing. All 500 training steps enter the displayed curves. The maximum ratio of the smoothed incorrect and correct mean-call curves is 1.90 at step 388. Period means and sign frequencies use raw values, weighting each step equally within this training run.

G.3 Tool-Token Density and Aggregate Credit

Token counts. For trajectory i, L_i counts valid generated response tokens across its turns, and m_i counts tokens in complete recognized tool-call spans, including delimiters. Token membership is determined by overlap between decoded character ranges and recognized tool-call spans. Both counts use the generated-response mask, with the tool mask additionally selecting the tokens included in m_i.

Pooled tool-token density. Within each training batch, the density for each correctness slice is its total tool-token count divided by its total generated-turn count. Figure 3(a) plots the raw batch-level correct-minus-incorrect difference. The mean batch-level gap is -24.35 tokens per turn over steps 1–200 and -121.90 over steps 201–500, a factor of 5.01 in magnitude. It is negative in 468 of 500 batches, including 295 of the 300 batches after step 200.

Batch tool credit. Figure 3(c) reports aggregate tool credit

$$\widehat{C}_{\text{tool}} = \frac{\sum_i m_i A_i}{\sum_i L_i},$$

where A_i is the advantage assigned by the training implementation after within-prompt reward normalization. Thus the numerator sums tool-token advantage coefficients and the denominator counts all generated response tokens. Among the 500 batches, 404 have negative credit and 96 have positive credit, giving a negative-credit frequency of 80.8%. The histogram uses bin width 0.004. The cumulative curve interpolates the empirical cumulative proportions at bin edges, zero, and plot bounds with a shape-preserving monotone cubic interpolation. The point at zero retains the exact negative-credit frequency.

Cross-dataset measurements. Table 5 contains the measurements plotted in Figure 3(b). The comparison varies the training dataset with the model, tool environment, training budget, and hyperparameters held fixed. Accuracy is training accuracy, and the gap is correct minus incorrect tool tokens per generated turn.

Table 5: Tool-token gap across mathematical training datasets. Accuracy is in %; the gap is correct minus incorrect tool tokens per turn.

Training dataset	Accuracy (%)	Tool-token gap
MATH	66.41	−151.55
DAPO-Math	30.59	−184.58
LIMR	78.13	−138.34
DeepScaleR	39.45	−127.04

G.4 Reward Means and Paired Credit Aggregation

Pooled reward means. Figure 8 computes reward means over all trajectories in each training batch. The trajectory-weighted mean is the fraction of correct trajectories; the turn-weighted mean is the fraction of generated turns contributed by correct trajectories. Over steps 201–500, their mean values are 0.251745 and 0.197714, respectively, a difference of 5.403 percentage points. The curves use EMA smoothing; interval means are computed from raw batch values.



Figure 8: Pooled reward means under trajectory and turn weighting. Incorrect trajectories contribute a larger share of generated turns than of sampled trajectories, lowering the turn-weighted mean.

Paired aggregation. Figure 4(a) applies token-mean and trajectory-mean aggregation to the same batches and their turn-level advantages, using Eq. (4). The curves use EMA smoothing. Over steps 201–500, the raw mean credits are -0.01704 and $+0.02323$, respectively. Trajectory-mean credit is positive while token-mean credit is negative in 259 of these 300 batches (86.3%). This paired calculation evaluates the aggregation-induced credit shift in Eq. (5) with the sampled trajectories and their advantages held fixed.

Paired credit signs. Figure 4(b) groups all 500 batches from steps 1–500 by the signs of their raw aggregate tool credit under the two aggregations. From trajectory mean to token mean, 368 batches (73.6%) change from positive to negative, and five (1.0%) change from negative to positive; 91 (18.2%) remain positive and 36 (7.2%) remain negative. The negative-credit frequency consequently rises from 8.2% under trajectory-mean aggregation to 80.8% under token-mean aggregation. Each connecting band represents the same batches evaluated under both weighting schemes, with its width proportional to their count.

G.5 Execution-Conditioned Continuation

Evaluation population. Figure 5(a,b) uses evaluation trajectories at step 500. The panels pool these trajectories. Calls are classified as successful or failed from tool feedback, with errors and timeouts included among failures. The comparison includes calls with an identified execution outcome.

Next-turn tool-call probability. For each execution status and call index, the numerator counts calls followed by a tool call in the next generated turn, and the denominator counts calls with an observed next generated turn. A subsequent tool call is identified by nonempty tool feedback. The call-index groups are 1, 2, 3, and $4+$, with the last group pooling indices 4-7. After the first execution, the next-turn call probabilities are $281/2360 = 11.9\%$ following success and $2200/2462 = 89.4\%$ following failure.

Continuation length. Figure 5(b) counts generated assistant tokens in turns strictly after the first tool-call turn, grouping trajectories by the first execution's status. The groups contain 2,360 trajectories following success and 2,462 following failure. Lengths are obtained by retokenizing the saved assistant response text with the Qwen2.5-3B-Instruct tokenizer, with special-token insertion, padding, and truncation disabled. The median lengths are 107 and 2,048 tokens, and the means are 473.17 and 2,326.64 tokens, respectively. The panel plots empirical cumulative distributions over the full continuation-length range. Its horizontal scale is linear through 500 tokens and logarithmic with base 2 thereafter.

G.6 Retry Statistics by Final Correctness

Table 6 gives retry rates and mean generated turns for four mathematics models and two agent environments, grouped by final task correctness. Figure 5(c) displays the four MATH/Python retry-rate pairs, with bar heights taken from the table and labels rounded to one decimal place.

Table 6: Reported retry rates and mean rollout turns, separated by terminal correctness. All four named models use MATH/Python; TACO and WebShop use Qwen2.5-1.5B-Instruct in their respective environments.

	Retry rate (%)		**Mean turns**	
Setting	Incorrect	Correct	Incorrect	Correct
Mathematical reasoning with Python				
Qwen2.5-1.5B-Instruct	31.38	12.07	2.20	1.91
Qwen2.5-3B-Instruct	60.55	35.96	3.15	2.38
Qwen2.5-7B-Instruct	36.82	21.48	1.76	1.59
Llama-3.2-1B-Instruct	88.65	50.88	4.05	3.37
Agent tasks with Qwen2.5-1.5B-Instruct				
TACO	77.03	58.68	2.96	2.44
WebShop	85.34	78.14	5.29	4.81

H TACA Method Details

This section develops the optimization properties of the advantage adjustments in Section 4 and specifies their integration with replay supervision.

H.1 Tool-Call Tokens and Decision Positions

For a prompt x, let $\mathcal{G}_x$ index its sampled trajectories, with terminal reward R_i and base GRPO advantage A_i for trajectory i. A generated token is indexed by (i, h, t), where h is the generated turn and t the position within that turn. The set $\mathcal{Z}$ contains the model-generated delimiters and content forming tool calls across $\mathcal{G}_x$. Preceding reasoning and returned tool observations provide conditioning context; the observations are excluded from the policy objective.

Let $\mathcal{T}_x \subseteq \mathcal{G}_x$ contain trajectories invoking at least one tool and $\mathcal{N}_x = \mathcal{G}_x \setminus \mathcal{T}_x$ those answering without tools. The set $\mathcal{Z}_{\mathrm{dec}} \subseteq \mathcal{Z}$ contains tokens encoding tool-call opening markers, such as `<python_code>`. The set $\mathcal{Z}_{\mathrm{dir}}$, disjoint from $\mathcal{Z}$, contains tokens encoding answer-opening markers, such as `<answer>`, from trajectories in $\mathcal{N}_x$. With the common tool-token offset $b_{x,\mathcal{Z}}$, reward contrast U_x, and contrast weight $\lambda_{\mathrm{dc}} \geq 0$, Eq. (9) becomes

$$A^{\mathrm{TACA}}_{i,h,t} = \begin{cases} A_i + b_{x,\mathcal{Z}} + \lambda_{\mathrm{dc}} U_x, & (i,h,t) \in \mathcal{Z}_{\mathrm{dec}}, \\ A_i + b_{x,\mathcal{Z}}, & (i,h,t) \in \mathcal{Z} \setminus \mathcal{Z}_{\mathrm{dec}}, \\ A_i - \lambda_{\mathrm{dc}} U_x, & (i,h,t) \in \mathcal{Z}_{\mathrm{dir}}, \\ A_i, & \text{all other generated tokens.} \end{cases} \tag{30}$$

Tool-call openings receive both adjustments, while the remaining tool-call tokens receive the same uniform offset. Each trajectory's base advantage remains present in every case.

H.2 Advantage Projection as a Minimal Adjustment

For a nonempty tool-call token set $\mathcal{Z}$, collect the base advantages into a vector $\boldsymbol{a} \in \mathbb{R}^{|\mathcal{Z}|}$, with one entry A_i for each token $(i, h, t) \in \mathcal{Z}$. Its mean is $\bar{A}_{x,\mathcal{Z}} = |\mathcal{Z}|^{-1}\mathbf{1}^\top \boldsymbol{a}$, where $\mathbf{1}$ is the all-ones vector. Let $\boldsymbol{v}$ denote a candidate tool-token advantage vector, permitting a distinct adjustment at every position.

Proposition H.1 (Minimum-distance tool-credit adjustment)**.** *The unique solution of*

$$\min_{\boldsymbol{v} \in \mathbb{R}^{|\mathcal{Z}|}} \frac{1}{2}\|\boldsymbol{v} - \boldsymbol{a}\|_2^2 \quad \textit{subject to} \quad \mathbf{1}^\top \boldsymbol{v} \geq 0 \tag{31}$$

is $\boldsymbol{a}^{\mathrm{proj}} = \boldsymbol{a} + b_{x,\mathcal{Z}}\mathbf{1}$*, with* $b_{x,\mathcal{Z}} = \max\{-\bar{A}_{x,\mathcal{Z}}, 0\}$ *as in Eq. (6).*

Proof. The objective is strictly convex and the constraint is a halfspace. Introducing the nonnegative Lagrange multiplier ν for $-\mathbf{1}^\top \boldsymbol{v} \leq 0$ gives the stationarity condition $\boldsymbol{v} = \boldsymbol{a} + \nu\mathbf{1}$. If $\bar{A}_{x,\mathcal{Z}} \geq 0$, the original vector is feasible and $\nu = 0$. Otherwise, complementary slackness requires $\mathbf{1}^\top \boldsymbol{v} = 0$, yielding $\nu = -\bar{A}_{x,\mathcal{Z}}$. Both cases give the stated solution, and strict convexity establishes uniqueness. □

The projection changes only the uniform component of the advantage vector:

$$\boldsymbol{a}^{\mathrm{proj}} - \max\{\bar{A}_{x,\mathcal{Z}}, 0\}\mathbf{1} = \boldsymbol{a} - \bar{A}_{x,\mathcal{Z}}\mathbf{1}. \tag{32}$$

Consequently, the centered variation and all pairwise advantage differences within the tool-call set are preserved. The squared adjustment is $|\mathcal{Z}|b^2_{x,\mathcal{Z}}$. Advantage Projection therefore satisfies the aggregate-credit constraint with the smallest total squared change to the sampled token advantages.

For the MT-GRPO advantages of Section 2, the correction also has a direct expression in terms of the imbalance analyzed in Section 3.2. For a nonempty tool-call set in a prompt group containing both binary reward outcomes, let M be the total generated-turn count, σ_{turn} the turn-weighted reward standard deviation, and $\bar{m}^+_{\mathrm{turn}}$ and $\bar{m}^-_{\mathrm{turn}}$ the tool-token counts per generated turn in correct and incorrect trajectories. Proposition 3.1 gives

$$b_{x,\mathcal{Z}} = \frac{M\sigma_{\mathrm{turn}}}{|\mathcal{Z}|}\max\{\bar{m}^-_{\mathrm{turn}} - \bar{m}^+_{\mathrm{turn}}, 0\}. \tag{33}$$

Thus the projection offset responds to the outcome-dependent tool-token imbalance through the same quantities that determine aggregate tool credit. Its general definition in Eq. (6) applies directly to scalar-reward advantages.

H.3 Shrinkage and Credit in Decision Contrast

When the tool-using and direct-answer trajectory sets $\mathcal{T}_x$ and $\mathcal{N}_x$ are both nonempty, define the empirical reward gap

$$d_x = \frac{1}{|\mathcal{T}_x|} \sum_{i \in \mathcal{T}_x} R_i - \frac{1}{|\mathcal{N}_x|} \sum_{i \in \mathcal{N}_x} R_i.$$

Each trajectory contributes once to its mode's mean. With $n_x = \min\{|\mathcal{T}_x|, |\mathcal{N}_x|\} > 0$ and shrinkage parameter $\eta \geq 0$, the contrast in Eq. (7) minimizes a regularized quadratic over a candidate scalar contrast u:

$$U_x = \operatorname*{argmin}_{u \in \mathbb{R}} \left\{ \frac{n_x}{2}(u - d_x)^2 + \frac{\eta}{2}u^2 \right\} = \frac{n_x}{n_x + \eta} d_x. \tag{34}$$

Differentiating the quadratic gives $(n_x + \eta)u - n_x d_x = 0$. The resulting contrast retains the sign of the empirical reward gap and approaches that gap as both groups grow for fixed η. When either mode is absent, $U_x = 0$.

The smaller group size determines the largest influence of an individual reward on this comparison. For fixed mode memberships,

$$\frac{\partial U_x}{\partial R_i} = \frac{n_x}{n_x + \eta} \begin{cases} |\mathcal{T}_x|^{-1}, & i \in \mathcal{T}_x, \\ -|\mathcal{N}_x|^{-1}, & i \in \mathcal{N}_x, \end{cases} \qquad \max_{i \in \mathcal{G}_x} \left| \frac{\partial U_x}{\partial R_i} \right| = \frac{1}{n_x + \eta}. \tag{35}$$

Hence changing one trajectory's reward by δ changes the contrast by at most $|\delta|/(n_x + \eta)$. This characterizes how shrinkage controls individual-sample influence when one response mode is sparsely represented.

For nonempty decision sets, let $\bar{A}_{x,\mathrm{dec}}$ and $\bar{A}_{x,\mathrm{dir}}$ be the means of the base token advantages on $\mathcal{Z}_{\mathrm{dec}}$ and $\mathcal{Z}_{\mathrm{dir}}$, respectively, and use the superscript TACA for their means after both adjustments. Averaging Eq. (30) yields

$$\bar{A}^{\mathrm{TACA}}_{x,\mathrm{dec}} - \bar{A}^{\mathrm{TACA}}_{x,\mathrm{dir}} = \bar{A}_{x,\mathrm{dec}} - \bar{A}_{x,\mathrm{dir}} + b_{x,\mathcal{Z}} + 2\lambda_{\mathrm{dc}} U_x. \tag{36}$$

The projection offset adjusts the tool side, and the contrast shifts the difference by $2\lambda_{\mathrm{dc}} U_x$ according to the empirical preference between response modes. Summing over all tool-call tokens gives the corresponding aggregate identity:

$$\sum_{(i,h,t) \in \mathcal{Z}} A^{\mathrm{TACA}}_{i,h,t} = |\mathcal{Z}| \max\{\bar{A}_{x,\mathcal{Z}}, 0\} + \lambda_{\mathrm{dc}} |\mathcal{Z}_{\mathrm{dec}}| U_x. \tag{37}$$

The two terms separate the aggregate-credit adjustment from the signed reward comparison at tool initiation. Within the same prompt group, tokens in the same case of Eq. (30) receive a common adjustment, preserving the base difference $A_i - A_j$ between trajectories i and j.

H.4 Supervised Replay in the Original Interaction Context

Supervised Replay stores tool-calling responses from trajectories with correct final outcomes, together with the histories under which they were generated. For each stored pair (c, y), c contains the prompt and interaction history preceding the response, including earlier tool observations, and y is the generated response containing the call. At each policy update, eligible context-response pairs are appended to a buffer of capacity B; entries are discarded in arrival order when the capacity is exceeded. A replay minibatch $\mathcal{D}_{\mathrm{rep}}$ is sampled uniformly from the buffer.

For a response y, the mask $m^{\mathrm{rep}}_t(c, y)$ equals one at its tool-call token positions and zero elsewhere. The number of supervised tokens is

$$M_{\mathrm{rep}} = \sum_{(c,y) \in \mathcal{D}_{\mathrm{rep}}} \sum_{t=1}^{|y|} m^{\mathrm{rep}}_t(c, y).$$

Under the current policy π_θ, teacher forcing evaluates each recorded tool-call token y_t conditional on c and the recorded response prefix $y_{<t}$. This prefix includes preceding reasoning in the same response. For $M_{\mathrm{rep}} > 0$, differentiating Eq. (8) gives the ascent contribution of the replay term:

$$-\lambda_{\mathrm{rep}} \nabla_\theta \mathcal{L}_{\mathrm{rep}} = \frac{\lambda_{\mathrm{rep}}}{M_{\mathrm{rep}}} \sum_{(c,y) \in \mathcal{D}_{\mathrm{rep}}} \sum_{t=1}^{|y|} m^{\mathrm{rep}}_t(c, y) \nabla_\theta \log \pi_\theta(y_t \mid c, y_{<t}), \tag{38}$$

Algorithm 1 TACA policy update

Input: Policy π_θ; prompt batch $\mathcal{X}$; group size G; replay buffer $\mathcal{R}$; $\eta, \lambda_{\mathrm{dc}}, \lambda_{\mathrm{rep}}$.

$\pi_{\theta_{\mathrm{old}}} \leftarrow \pi_\theta$
for $x \in \mathcal{X}$ **do**
  Sample $\{\tau_i\}_{i \in \mathcal{G}_x} \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid x)$, with $|\mathcal{G}_x| = G$
  Compute terminal rewards R_i and advantages $A_i = A_i^{\mathrm{turn}}$ (Section 2)
  Form tool-using and direct-answer trajectory sets $\mathcal{T}_x$ and $\mathcal{N}_x$
  Identify tool-call tokens $\mathcal{Z}$ and decision tokens $\mathcal{Z}_{\mathrm{dec}}, \mathcal{Z}_{\mathrm{dir}}$
  // Advantage Projection: aggregate tool-credit adjustment
  $b_{x,\mathcal{Z}} \leftarrow \max\{-\bar{A}_{x,\mathcal{Z}}, 0\}$ if $\mathcal{Z} \neq \varnothing$, else 0 Eq. (6)
  // Decision Contrast: tool-use versus direct-answer rewards
  $U_x \leftarrow \mathrm{Decision\ Contrast}(\mathcal{T}_x, \mathcal{N}_x, \{R_i\}_{i \in \mathcal{G}_x}; \eta)$ Eq. (7)
  $A_{i,h,t}^{\mathrm{TACA}} \leftarrow A_i + b_{x,\mathcal{Z}} \mathbf{1}\{(i,h,t) \in \mathcal{Z}\} + \lambda_{\mathrm{dc}} U_x \left[\mathbf{1}\{(i,h,t) \in \mathcal{Z}_{\mathrm{dec}}\} - \mathbf{1}\{(i,h,t) \in \mathcal{Z}_{\mathrm{dir}}\}\right]$
end for
// Supervised Replay: replay supervision for tool-call generation
Update $\mathcal{R}$ with context-response pairs (c, y) from tool-calling turns in sampled trajectories with correct final outcomes
Uniformly sample a replay minibatch $\mathcal{D}_{\mathrm{rep}}$ from $\mathcal{R}$
$\mathcal{L}_{\mathrm{rep}}(\theta) \leftarrow \mathrm{Supervised\ Replay}(\pi_\theta, \mathcal{D}_{\mathrm{rep}})$ Eq. (8)
// Policy update with transition-aware advantages and replay supervision
Update π_θ by maximizing $\mathcal{J}_{\mathrm{TACA}}(\pi_\theta)$ with $A_{i,h,t}^{\mathrm{TACA}}$ and $\mathcal{L}_{\mathrm{rep}}(\theta)$ Eq. (10)

Output: Updated policy π_θ and replay buffer $\mathcal{R}$.

where $\lambda_{\mathrm{rep}} \geq 0$ is the replay weight. Each tool-call token in the replay minibatch contributes a log-likelihood gradient with coefficient $\lambda_{\mathrm{rep}}/M_{\mathrm{rep}}$, conditioned on its original interaction context and preceding response tokens. The retained responses from correct trajectories provide tool-generation targets across successive policy updates. An empty replay minibatch contributes zero loss.

H.5 Training Procedure

Algorithm 1 summarizes one TACA update through the three component operations defined in Section 4. The set $\mathcal{G}_x$ indexes the G trajectories sampled for prompt x. Empty tool-call sets and replay minibatches contribute zero offset and zero replay loss, respectively; the reward contrast is zero when either trajectory group is empty.

I Implementation Details

I.1 Training Configuration

Models and interaction environment. We post-train Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct [38], and Llama-3.2-1B-Instruct on MATH [13]. Multi-turn training uses AlphaApollo [61] on four A800 GPUs, with Python code execution as the primary reasoning tool. Each rollout alternates generated responses with tool observations until a final answer or the six-turn limit. Tool observations provide context; policy updates score model-generated tokens.

Sampling and optimization. Each training batch contains 32 prompts and eight trajectories per prompt, yielding 256 trajectories. Rollouts use temperature 1.0 and top-p 1.0. We optimize with AdamW at a constant learning rate of 10^{-6}, with one policy-optimization epoch per rollout batch. The implementation applies dual clipping with coefficient 3.0 to negative-advantage updates. Table 7 gives the sampling budgets and optimization parameters. The response budget is 1,024 tokens per turn for Qwen2.5-7B-Instruct and 2,048 for the other backbones. The policy-training seed is 42.

TACA configuration and ablations. The replay buffer stores tool-calling responses from trajectories with correct final outcomes, together with their preceding interaction contexts, and supplies up to 32 pairs per policy update, sampled uniformly. Table 7 specifies the fixed weights for the reward contrast and replay loss, together with the contrast shrinkage and replay capacity. The ablations in Table 3 disable the Advantage Projection offset, Decision Contrast term, or Supervised Replay loss individually.

Table 7: Configuration for the mathematical reasoning experiments. TACA coefficients are shared across the evaluated backbones. Replay sizes count stored context-response pairs.

Parameter	**Setting**
Rollout configuration	
Prompts per batch / trajectories per prompt	32 / 8
Training temperature / top-p	1.0 / 1.0
Maximum input-context length	7,168 tokens
Maximum generated turns	6
Response tokens per turn	1,024 (Qwen2.5-7B); 2,048 (other models)
Python execution timeout	3 seconds
Policy optimization	
Optimizer / learning rate	AdamW / 10^{-6}
Learning-rate schedule / warm-up	Constant / 0 steps
Adam coefficients / weight decay	$(0.9, 0.999)$ / 0.01
Gradient-norm clipping	1.0
Policy epochs per rollout batch	1
Importance-ratio clipping threshold ε	0.2
Negative-advantage dual-clipping parameter	3.0
KL weight β / entropy coefficient	0 / 0
Policy-training seed	42

I.2 Tasks, Rewards, and Evaluation

Mathematical reasoning benchmarks. The evaluation suite comprises six benchmarks: MATH500 [13], a 500-problem subset of the MATH test set; AIME 2024[2] and AIME 2025[3], containing short-answer competition problems; AMC 2023[4], containing problems adapted from AMC 12A and AMC 12B in numerical-answer form; OlympiadBench [12], covering olympiad-level mathematical reasoning; and GSM8K [2], covering multi-step arithmetic word problems.

Answer verification and evaluation sampling. We use terminal answer correctness as the mathematical reasoning reward [56]. For trajectory i, let a_i denote its extracted final answer and $a^\star$ the reference answer. Math-Verify[5] checks their mathematical equivalence, assigning terminal reward $R_i = 1$ for an equivalent answer and $R_i = 0$ otherwise. Evaluation samples 32 responses per question at temperature 0.6 and top-p 1.0. Benchmark scores report mean correctness over sampled responses as a percentage; the overall score is their unweighted six-benchmark mean.

Coding. We train Qwen2.5-1.5B-Instruct on TACO [20] and evaluate on its test set. The coding environment provides a public-test runner and a scratch Python executor, with unit-test outcomes determining terminal rewards. Table 2 reports unit-test scores separately for HackerRank, HackerEarth, CodeForces, and CodeChef; the TACO average weights the four sources equally.

Interactive decision-making. We train Qwen2.5-1.5B-Instruct separately on WebShop [52] using its official data split. The policy interacts through search, navigation, and purchase actions and receives a terminal task reward. Evaluation reports success rate for completing the shopping goal and the task score supplied by the environment, as shown in Table 2.

Baseline comparisons. Base-model inference and agentic prompting evaluate the starting models without policy updates. The trained baselines comprise non-agentic GRPO [34], ReTool [5], MT-GRPO [6], GiGPO [6], and SimpleTIR [51]. The mathematical reasoning comparison uses the

[2] https://huggingface.co/datasets/HuggingFaceH4/aime_2024
[3] https://huggingface.co/datasets/opencompass/AIME2025
[4] https://huggingface.co/datasets/math-ai/amc23
[5] https://github.com/huggingface/Math-Verify

six-benchmark protocol above; the coding and interactive comparisons use the corresponding TACO and WebShop metrics. Training and evaluation settings are matched within each model-task pair.

I.3 Interaction Prompts

The agentic mathematical reasoning experiments use the templates in Figures 9 and 10 for training and evaluation. The first-turn template presents the question and action format; later turns additionally receive the generated-turn count and interaction history. Responses place reasoning inside `<think>` tags and select either a Python call delimited by `<python_code>` or a final answer delimited by `<answer>`. Execution results are returned as tool observations for the next turn.

Prompt Template (No History)

```
You are a problem solver agent that combines step-by-step reasoning with Python code
execution to solve problems accurately.

Your question: {question}

You MUST first think inside <think> </think> tags, then you MUST choose exactly ONE of
the following actions:
1) [RECOMMENDED] <python_code>...</python_code>: Write exactly  ONE block of pure
    Python 3 to compute, verify, or explore the problem. You SHOULD use this tool
    to check your reasoning before giving a final answer. Inspect the <tool_response>
    (stdout from your code). If it disagrees with your reasoning, correct yourself.
2) <answer>...</answer>: Provide your final answer ONLY after you have verified it
    (ideally via code). You MUST format the answer in LaTeX inside \boxed{...}.

IMPORTANT RULES:
- You MUST use <python_code> at least once to verify your reasoning before providing
your final <answer>. Skipping code verification leads to errors.
- You MUST NOT output both <python_code> and <answer> in the same step.
- Your <think> block MUST come before any action tag.
```

Figure 9: Prompt template used for the first step (no interaction history).

Prompt Template (With History)

```
You are a problem solver agent that combines step-by-step reasoning with Python code
execution to solve problems accurately.

Your question: {question}

Prior to this step, you have already taken {step_count} step(s).
Below is the interaction history:
{memory_context}

1) [RECOMMENDED] <python_code>...</python_code>: Write exactly  ONE block of pure
    Python 3 to compute, verify, or explore the problem. You SHOULD use this tool
    to check your reasoning before giving a final answer. Inspect the <tool_response>
    (stdout from your code). If it disagrees with your reasoning, correct yourself.
2) <answer>...</answer>: Provide your final answer ONLY after you have verified it
    (ideally via code). You MUST format the answer in LaTeX inside \boxed{...}.

IMPORTANT RULES:
- You MUST use <python_code> at least once to verify your reasoning before providing
your final <answer>. Skipping code verification leads to errors.
- You MUST NOT output both <python_code> and <answer> in the same step.
- Your <think> block MUST come before any action tag.
```

Figure 10: Prompt template used for subsequent steps (with interaction history).

Table 8: Comparison with Dr. GRPO on Qwen2.5-1.5B-Instruct (accuracy, in %).

Method	MATH500	AIME24	AIME25	AMC23	Olympiad Bench	GSM8K	Avg.
MT-GRPO	50.70	3.26	0.97	**27.50**	17.88	71.49	28.63
Dr. GRPO	51.76	4.03	0.89	27.40	17.57	72.56	29.04
TACA	**59.14**	**8.43**	**8.12**	27.42	**22.20**	**74.15**	**33.24**

Table 9: Unsmoothed training statistics over steps 1–100 on Qwen2.5-1.5B-Instruct.

Metric	ReTool	TACA
Training accuracy (%)	49.17	48.09
Tool-use rate (%)	1.49	75.76
No-tool failure rate (%)	49.75	18.15
Tool-using correct trajectories (%)	0.41	42.00
Tool use among correct trajectories (%)	0.84	87.34
Steps containing calls	27	100
Attempted calls	730	54,924
Successful calls	580	47,391
Execution success (%)	79.45	86.28

I.4 Normalization Comparison

Comparison protocol. We compare MT-GRPO, Dr. GRPO [25], and TACA on Qwen2.5-1.5B-Instruct using the same MATH training data, Python environment, and six-benchmark evaluation protocol. Dr. GRPO uses a constant response normalizer and omits reward-standard-deviation normalization. TACA uses the adjusted advantages and replay loss in Eq. (10).

Results. Table 8 gives mean accuracies of 28.63%, 29.04%, and 33.24% for MT-GRPO, Dr. GRPO, and TACA. TACA gains 4.20 points over Dr. GRPO and improves all six benchmarks.

I.5 Training Diagnostics

Trajectory and execution statistics. Table 9 summarizes unsmoothed records for Qwen2.5-1.5B-Instruct on MATH over steps 1–100, comprising 25,600 trajectories per method. Training accuracy, tool-use rate, and no-tool failure rate are the fractions of trajectories that answer correctly, invoke a tool, and answer incorrectly without tools, respectively. Tool-using correct trajectories are reported both as a fraction of all trajectories and as a fraction of correct trajectories. Execution success divides the total successful calls by the total attempted calls over the same window. For rates over all trajectories, the fixed batch size makes per-step and pooled averages equivalent.

Curve construction. Figure 7 displays training steps 1–100 with exponential moving average (EMA) smoothing. Panels (a–c) show the per-step trajectory rates defined above. Panel (d) shows the per-step execution-success rate, with zero assigned to steps without tool calls. The pooled execution rate in Table 9 is the ratio of total successful calls to total attempted calls over the displayed window.

J NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have summarized the position and key contributions of the paper in the abstract and introduction parts.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide complete proofs in Appendices F and H.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experiment details of implementation are introduced in Section 5 and Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have open-sourced our code at https://github.com/tmlr-group/TACA.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experiment details of implementation are introduced in Section 5 and Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We conduct evaluations and repeat the inference for each question to obtain reliable results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computed requirements are introduced in Section 5 and Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully checked the NeurIPS Code of Ethics and confirmed that our paper obeys it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts are introduced in Appendix C.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release new models or datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have introduced the resources of the models and datasets used in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This work does not introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper is not about crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper is not about research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this work does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.