Unveiling Entropy-Performance Decoupling in Agentic RL for Tool-Integrated Reasoning

Yirong Zeng[♠,‡] **Shen You**[♠] **Yufei Liu**[♡] **Hao Cong**[♣]
Yutai Hou[◇] Xiao Ding[♠,*] Yuxian Wang[◇] Wu Ning[◇] Wang Xu[◇]
Bibo Cai[♠,*] Dandan Tu[◇] Wenyu Zang[♦] Ting Liu[♠]

[♠]Harbin Institute of Technology, SCIR Lab
[♡]Peking University, [♣]Tsinghua University
[◇]Huawei Technologies Co., Ltd. , [♦]Academy of Cyber, China
[‡]Work done while interning at Huawei.

Abstract

Tool-Integrated Reasoning (TIR) enables LLMs to solve complex, multi-step problems by offloading computation to external tools. While Agentic Reinforcement Learning (RL) has advanced TIR systems, the underlying policy entropy dynamics governing these models remain poorly understood. In this paper, we conduct an extensive empirical study and uncover a pervasive phenomenon of entropy–performance decoupling: in the late stages of RL training, policy entropy continues to diverge despite performance stagnation. Moreover, this ineffective entropy growth is primarily driven by an accumulation of invalid tool-use trajectories, which trap the agent in non-informative hallucination loops and degrade final reasoning accuracy. To address this bottleneck, we propose EarlyTIR, a strategic truncation mechanism that prunes non-informative exploration during the RL rollout phase. EarlyTIR halts trajectories that exhibit excessive invalid interactions while preserving the model's ability to learn essential error-recovery behaviors. Empirical evaluations across six public benchmarks and two model families demonstrate that EarlyTIR consistently elevates the performance ceiling, achieving a macro-average accuracy boost of 13.6% on the Qwen3-32B model. Furthermore, our approach enhances inference efficiency, reaching correct solutions with 33.7% fewer tool interactions compared to strong baselines [2].

1 Introduction

Large Language Models (LLMs) have achieved remarkable success in natural language understanding, yet they remain fundamentally constrained by their internal reasoning capacity when facing complex, multi-step problems. Tool-Integrated Reasoning (TIR) has emerged as a critical paradigm to overcome these boundaries, enabling models to offload symbolic computation and logical verification to external executors such as code interpreters and APIs [16, 13]. TIR empowers LLMs to navigate domains that were previously inaccessible through text-only reasoning, such as advanced software engineering and formal mathematical proof.

The frontier of TIR research has recently shifted toward Agentic Reinforcement Learning (Agentic RL), where models refine their reasoning and tool-use strategies through multi-turn interactions with the environment [35]. Current efforts primarily focus on engineering verifiable reward functions or optimizing action spaces [5, 10]. However, a fundamental but neglected driver of this learning process

*Corresponding authors: {xding,bbcai}@ir.hit.edu.cn.
[2]Our code is available at https://github.com/zeng-yirong/EarlyTIR.

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

is entropy dynamics, the evolution of policy uncertainty throughout training. While entropy is known to dictate the performance ceiling in text-only LLMs [4], its role in TIR remains an unexplored black box. Crucially, TIR entropy is distinct: it is shaped by discrete, external information gains from environment feedback, a mechanism entirely absent in traditional generative tasks [11]. This lack of understanding leaves a critical gap in our ability to stabilize training and push the boundaries of agentic intelligence.

In this work, we first conduct an extensive empirical pre-study across diverse model architectures to demystify these dynamics. Our analysis reveals a pervasive entropy–performance decoupling phenomenon that challenges the traditional assumption of monotonic entropy decay during convergence. As illustrated in Figure 1, early-stage training exhibits a healthy, synchronized growth of entropy and accuracy, reflecting the agent's active exploration of the hybrid action space. The later stages, however, suffer from a detrimental shift: performance plateaus while policy entropy continues to diverge. This indicates that a substantial portion of policy uncertainty in late-stage RL becomes ineffective, failing to yield further gains in reasoning precision.

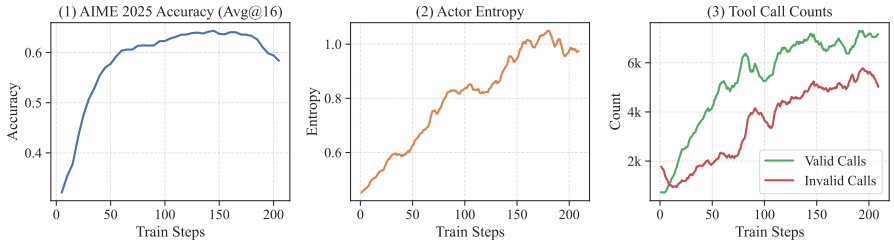


Figure 1: Entropy dynamics and invalid/valid tool-call counts in Qwen3-32B agentic RL training. We observe that while accuracy reaches a plateau, policy entropy continues to diverge in later training stages. This ineffective entropy is driven by a significant surge in invalid tool calls, which fail to provide informative environment feedback.

Further investigation identifies the accumulation of invalid tool-use trajectories as the primary driver of this ineffective entropy. Our analysis demonstrates a strong positive correlation between the ratio of invalid tool calls (such as execution errors and redundant invocations) and terminal policy uncertainty. Specifically, we observe that as reasoning trajectories extend, models often fall into an error-correction trap: high-entropy states trigger a cascade of non-informative environment interactions, leading to hallucination loops that precipitously degrade final accuracy. These insights suggest that the Agentic RL paradigm is bottlenecked by the model's inability to suppress non-informative feedback, trapping the agent in high-uncertainty but low-utility reasoning paths.

To address these challenges, we propose EarlyTIR (Early intervention for Tool-Integrated Reasoning), a strategic truncation mechanism designed to prune non-informative trajectories during the RL rollout phase. By isolating noisy feedback from the model's context window and assigning targeted penalties to these truncated paths, EarlyTIR effectively breaks the hallucination loop and refocuses the training objective on high-utility reasoning. Empirical evaluations across six public benchmarks and two model families (Qwen2.5 and Qwen3, 7B to 32B) demonstrate that EarlyTIR consistently boosts reasoning accuracy, achieving a macro-average improvement of 13.6% on the Qwen3-32B model, while effectively suppressing ineffective entropy growth. Furthermore, EarlyTIR-trained models demonstrate superior inference efficiency, reaching correct solutions with 33.7% fewer tool interactions.

2 Preliminaries

2.1 Problem Formulation

We formulate the tool-integrated reasoning process as a partially observable Markov decision process, denoted as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \Omega, O \rangle$. Since the underlying environment state $s_t \in \mathcal{S}$ is not fully accessible, the agent maintains an observation history $h_t = (o_0, a_0, r_0, \ldots, o_t) \in \mathcal{H}$ to track the trajectory. At each time step t, the agent samples an action $a_t \sim \pi_\theta(a|h_t)$ from a hybrid action space $\mathcal{A}$ according to its stochastic policy π_θ, where the policy entropy $H(\pi_\theta(\cdot|h_t))$ characterizes the uncertainty of decision-making.

Hybrid Action Space. Distinct from standard autoregressive language modeling, the TIR action space $\mathcal{A} = \mathcal{A}_{\text{gen}} \cup \mathcal{A}_{\text{tool}}$ is partitioned into two functional sets:

- **Generation Actions ($\mathcal{A}_{\text{gen}}$):** These correspond to tokens from the vocabulary $\mathcal{V}$ used for internal Chain-of-Thought (CoT) reasoning or for synthesizing the final answer. These actions primarily involve the internal state transitions of the LLM.
- **Tool Actions ($\mathcal{A}_{\text{tool}}$):** These represent structured interactions with external environments (e.g., Python interpreters). An action $a_t \in \mathcal{A}_{\text{tool}}$ typically manifests as a specialized token sequence forming a function call $c_t = (\text{func_name}, \text{args})$. Upon execution, the environment returns a new observation o_{t+1} (e.g., execution logs or error messages), providing discrete information gains to the agent.

2.2 Policy Entropy in TIR

In the context of Tool-Integrated Reasoning, policy entropy serves as a quantitative metric for the agent's decision-making uncertainty and exploration breadth. Given the stochastic policy π_θ defined over the hybrid action space $\mathcal{A}$, the entropy at time step t for a specific history h_t is defined as:

$$H(\pi_\theta(\cdot|h_t)) = -\sum_{a \in \mathcal{A}} \pi_\theta(a|h_t) \log \pi_\theta(a|h_t)$$

The average policy entropy over a trajectory ξ of length T is denoted as $\bar{H} = \frac{1}{T}\sum_{t=1}^{T} H(\pi_\theta(\cdot|h_t))$.

Token-Level Computation. Operationally, we compute the entropy at the token level: at each generated token, we evaluate the full-vocabulary distribution $\pi_\theta(\cdot \mid h_t)$ over $\mathcal{V}$ and aggregate the resulting token entropies over the trajectory. Tool-output tokens returned by the environment are masked from both the entropy calculation and the policy gradient, since they are not sampled from the policy. The action-level formulation above is therefore conceptual; the reported curves reflect the uncertainty of the agent's own generated tokens, including both reasoning text and tool-call tokens.

Entropy Mechanism in Text-Only Reasoning. Existing literature on RL for LLMs suggests that the performance ceiling is inextricably linked to policy entropy [4, 24]. During RLVR training on text-only reasoning tasks, a fundamental phenomenon has been observed: policy performance improves at the expense of policy entropy [7, 33]. Specifically, as the model converges toward higher rewards, policy entropy typically undergoes an exponential decay [4]. This relationship implies that while reducing uncertainty is necessary for accuracy, a precipitous drop in entropy often signals the loss of logical flexibility, leading to training collapse. In this work, we investigate whether this text-only entropy mechanism still holds when external tool feedback introduces new information gains.

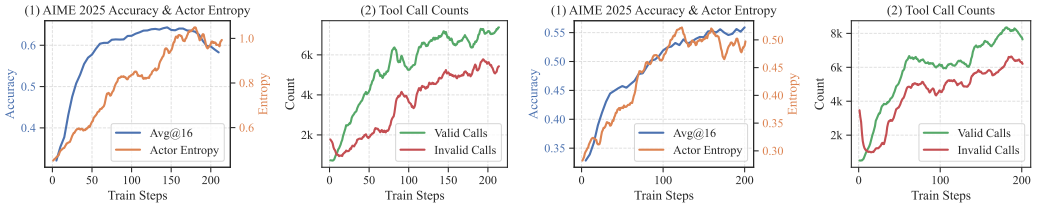
2.3 RLVR Algorithms

Reinforcement Learning with Verifiable Rewards (RLVR) has established itself as a paradigm for endowing LLMs with robust tool-integrated reasoning capabilities. By optimizing against outcome-oriented signals, RLVR enables models to explore and consolidate complex decision-making paths that are often missed in supervised learning. In this work, we adopt the DAPO algorithm [32], a state-of-the-art RL framework designed for large-scale reasoning tasks. DAPO enhances the standard Group Relative Policy Optimization (GRPO [7]) by introducing four key techniques, including *Decoupled Clipping* and *Dynamic Sampling*, which effectively mitigate entropy collapse and stabilize policy updates in long-CoT scenarios. The algorithm details are provided in Appendix F.1, and the hyperparameter settings in Appendix G.

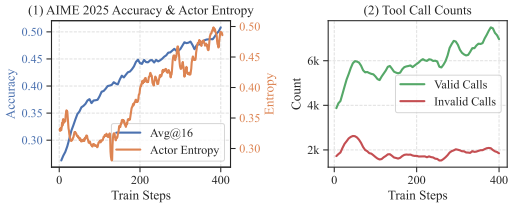
3 Pre-study: Entropy Dynamics Mechanism

3.1 Settings

We implement a unified experimental protocol encompassing two model families and five open-source base models, ranging from 7B to 32B parameters, together with two API-based LLMs for



(a) Qwen3-32B (Math). (b) Qwen3-8B (Math)



(c) Qwen2.5-32B-Instruct (Math)

Figure 2: Entropy dynamics and invalid/valid tool-call counts across models and tasks. All models exhibit a surge in invalid tool calls and decoupled entropy growth in later training stages, despite accuracy remaining stable.

inference-only evaluation. Our evaluation spans verifiable domains in mathematics and programming across six public benchmarks.

Models. For RL training, we utilize five widely adopted open-source base models from two distinct families. These include the Qwen2.5 series (7B and 32B [28]) and the Qwen3 series (8B, 14B, and 32B [29]). For inference-only evaluation, we further include two API-based LLMs to provide a broader competitive baseline: DeepSeek-V3.2 [7], Kimi-2.5 [22].

Tasks and datasets. We primarily focus on mathematics and coding tasks where rewards can be strictly verified. To ensure evaluation integrity, we maintain consistency between training tasks and downstream evaluation metrics.

- **Mathematics**: RL training is conducted on the *DAPO-Math-17k*[3] dataset, while Supervised Fine-Tuning (SFT) utilizes *ReTool-SFT*[4] from ReTool [5]. Evaluation is performed on AMO-Bench [1], HMMT 2025 [8], AIME 2024, and AIME 2025 [17].
- **Coding**: For RL training, we utilize the *DeepCoder* dataset[5] [14]. Models are evaluated on HumanEval [2] and LiveCodeBench (v6) [9].

RL training. Because base models differ in their initial reasoning and tool-use capabilities, we adopt a two-stage strategy to ensure a fair comparison. We follow the experimental setup of ReTool, implemented with the veRL framework [21]. For the Qwen2.5 family, we perform an initial SFT phase followed by RL refinement. Conversely, for the Qwen3 family, we apply RL directly to the base models. We employ DAPO [32] as our primary RL algorithm. We use a Python interpreter as the tool, with code execution hosted in a local sandbox for security and reproducibility. The pre-study uses a rollout size of 8 and a maximum horizon of 16 turns. See Appendix G for further hyperparameter details.

3.2 A First Glance

To investigate the entropy dynamics unique to TIR, we conduct four independent RL runs across various model architectures. As shown in Figures 1 and 2, these runs reveal a unique entropy–performance relationship in TIR that contradicts the trade-off typically observed in text-only tasks. Initially, we observe a simultaneous increase in policy entropy and performance, suggesting that

[3] https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k

[4] https://huggingface.co/datasets/JoeYing/ReTool-SFT

[5] https://huggingface.co/datasets/agentica-org/DeepCoder-Preview-Dataset

tool use triggers a healthy entropy-expansion phase during early training (e.g., < 50 steps). Invalid tool calls are also far fewer than valid ones, indicating that the model can effectively use tools to obtain information gain early in training. This pattern indicates that the model actively explores a broader action space to accommodate environment interactions. However, a decoupling occurs in later stages: performance remains stable while entropy continues to rise, rendering this additional uncertainty ineffective. Meanwhile, invalid tool calls grow substantially, indicating that the model falls into invalid-call loops and can no longer obtain useful information from the environment.

We also observe that for Qwen2.5-32B-Instruct (Figure 2c), invalid tool calls remain low while accuracy continues to improve. These observations imply that not all entropy contributes equally to the agent's reasoning capability. Entropy increases associated with invalid tool calls are generally ineffective and do not improve model performance. This discovery motivates us to distinguish between these two states. By encouraging effective exploration while suppressing ineffective entropy, we aim to overcome performance stagnation and raise the capability ceiling of Agentic RL.

3.3 Understanding Entropy and Tool Call Quality

In this section, we quantify how tool-call quality influences the entropy of multi-turn TIR [11]. We define invalid tool calls as those yielding no information gain (e.g., execution errors, timeouts, empty returns, or redundancies). We then examine how the invalid-call ratio affects entropy dynamics, as illustrated in Figure 3. We observe a strong positive correlation between the ratio of invalid calls and policy entropy. Entropy grows significantly faster under high invalidity ratios than under low ones. A Spearman's rank correlation analysis confirms this trend, showing a significant positive association between invalidity bins and terminal entropy levels ($\rho = 0.93, p < 0.05$). Notably, this divergence widens in later training stages, suggesting that invalid tool calls may be a primary driver of decoupled entropy growth.

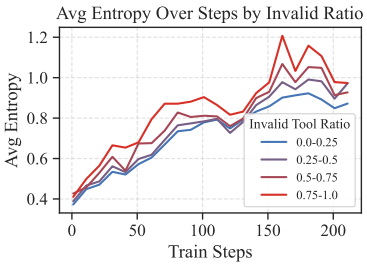
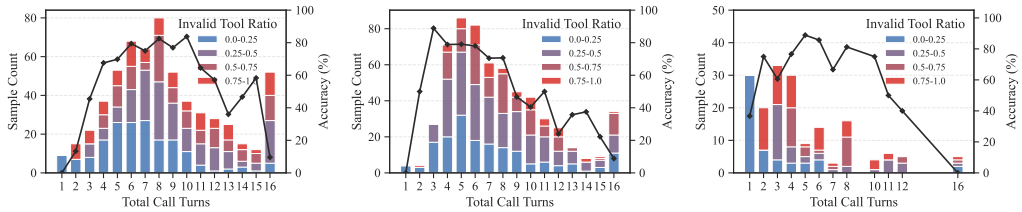


Figure 3: Entropy dynamics across different tool-call invalidity ratios for Qwen3-32B. Trajectories are partitioned into four bins based on ratio.



(a) RL-trained Qwen3-32B (Math) (b) RL-trained Qwen3-8B (Math) (c) Qwen3-32B (Math)

Figure 4: Analysis of reasoning accuracy vs. tool-call turns. The stacked bars represent the sample count per turn, color-coded by the *Invalid Tool Ratio*. The black line plots the accuracy (%) for each bin. High invalidity and long trajectories collectively contribute to the performance drop across all tested models.

We further quantify the evolution of invalid tool-call counts with RL training steps across multiple models (Qwen3-32B/8B and Qwen2.5-32B-Instruct), as shown in Figure 2 (additional results in Section H). Early in training, invalid calls decline slightly, accompanied by synchronized gains in accuracy and policy entropy. In contrast, during later stages, invalid calls surge despite plateauing accuracy, resulting in a decoupling where entropy grows without corresponding performance improvements. This pattern persists across model scales (32B vs. 8B) and architectures (Qwen3 vs. Qwen2.5). This suggests that invalid tool calls do not provide the information gain needed to resolve policy uncertainty. The agent is thus trapped in a high-entropy state that does not translate into performance gains. Notably, for Qwen2.5-32B-Instruct, both accuracy and entropy continue to rise after 4,000 training steps, while the invalid tool-call count remains consistently low. This empirical evidence confirms that, in the later stages of training, ineffective

entropy is largely driven by the accumulation of non-informative environmental interactions. We next examine how this ineffective entropy affects reasoning accuracy.

3.4 Impact of Tool-use Trajectory and Validity on Accuracy

To further understand the failure modes in TIR, we investigate how tool-use turns and their validity relate to final reasoning accuracy. We evaluate this across various architectures, including base models, RL-trained models, and commercial API-based LLMs (details in section H).

As illustrated in Figure 4, a consistent pattern emerges: reasoning accuracy is inversely correlated with the number of tool-call turns. Models achieve peak performance within a comfortable range of 3–6 turns, and accuracy degrades significantly once trajectories extend beyond 10 turns. This performance decay is closely linked to the invalid tool ratio (indicated by the color-coded stacked bars). In the shortest trajectories (e.g., 1–2 turns), tool use is almost entirely valid, yet accuracy remains sub-optimal, likely due to insufficient problem decomposition. In the optimal 3–6 turn range, most tool calls remain valid (blue segments), reflecting an efficient and focused reasoning process. In longer horizons, however, high-invalidity segments expand substantially (red segments, where $> 75\%$ of calls fail). This indicates that once a model fails to resolve a sub-problem within a few attempts, it tends to fall into a hallucination loop. It then repeatedly invokes invalid tools, parameters, or APIs, and eventually fails to produce the correct answer.

These findings underscore that the robustness of tool-augmented reasoning depends not merely on a model's capacity to invoke tools, but critically on its ability to maintain a coherent and valid tool-use trajectory.

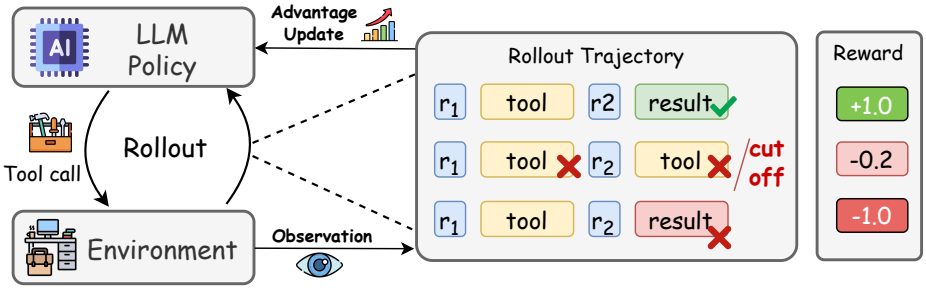


Figure 5: Overview of the proposed RL rollout intervention mechanism in TIR. We monitor invalid tool calls; a rollout whose cumulative invalid-call count exceeds the predefined threshold is immediately terminated and receives a negative reward.

4 Methodology

Given that invalid tool calls are the primary drivers of performance stagnation and ineffective entropy growth, simply masking or penalizing all invalid calls might seem intuitive. However, as analyzed in Section 3.4, trajectories with extremely short horizons (e.g., < 2 turns) yield suboptimal accuracy. In contrast, peak performance is consistently achieved within a comfortable range of interactions, which naturally tolerates a minor degree of invalid exploration. Therefore, a blanket prohibition is not appropriate. We need a more robust intervention mechanism that suppresses excessive invalid interactions while preserving beneficial exploration, thereby raising the capability ceiling of RL.

To this end, we introduce EarlyTIR (Early intervention for Tool-Integrated Reasoning), an early truncation mechanism for tool-call control in agentic RL algorithms, illustrated in Figure 5. During the RL rollout phase, EarlyTIR continuously monitors tool-use trajectories and employs a rule-based detector to identify four objectively verifiable failure modes: *execution errors* (non-empty standard error or non-zero exit code), *timeouts* (executor timer interrupt), *empty returns* (missing, `None`, or empty standard output), and *redundant commands* (exact match with the immediately preceding call). These signals are recorded by the execution sandbox rather than inferred by an LLM; Appendix E.3 reports a 50-trajectory audit with 100% precision and discusses the false-positive risk.

Instead of allowing the agent to accumulate unbounded errors, we introduce a tolerance threshold τ. If the cumulative number of invalid calls in a trajectory exceeds τ, EarlyTIR immediately halts generation, assigns the sample a penalizing reward ($R_t < 0$), and uses it as a negative sample in the DAPO loss.

By bounding the number of allowable errors, EarlyTIR prevents the model from falling into infinite hallucination or error-retry loops. Setting $\tau > 0$ also preserves the model's opportunity to learn how to recover from occasional early mistakes.

5 Experiments

5.1 Setup

Implementation Details. In our RL training pipeline, we sample a rollout batch of 512 prompts with a group size of 16, and use a PPO mini-batch of 128 prompts for each policy update. The generation process is capped at a maximum response length of 16K tokens and eight tool-call turns per trajectory. For the EarlyTIR mechanism, we empirically set the invalid tool-call tolerance threshold to $\tau = 2$ and apply an early truncation penalty reward of $R_t = -0.2$. All EarlyTIR results in Table 1 use this configuration. Other RL hyperparameters and environment configurations are summarized in Table 7 (Appendix G).

Baselines. In addition to prominent open-source and API-based LLMs, we compare our approach against recent RLVR-trained models in the TIR domain. For mathematical reasoning, our baselines include ReTool [5], SimpleTIR [27], and ZeroTIR-7B [15]. Notably, we re-implement ReTool on all evaluated base models to serve as a unified baseline. For code generation, we evaluate against DeepCoder [14], KodCode [26], and MicroCoder [12]. To ensure a fair comparison, all code-related baselines are trained using the same foundation model and the standard RLVR algorithm.

Table 1: Performance comparison on mathematical reasoning benchmarks. ♣ denotes base/instruction-tuned models, and ♡ denotes RL-trained models in the TIR domain. The **Average** column reports the macro-average across all datasets. EarlyTIR numbers are means over 3 independent runs.

Model	AIME24	AIME25	Hmmt25	AMO-Bench	Average
♣ Qwen3-8B	63.33	50.33	43.33	4.60	40.40
♡ ReTool	64.17	57.08	44.76	5.75	42.94
♡ **EarlyTIR-8B**	**67.08**↑3.75	**60.42**↑10.09	**46.67**↑3.34	**7.75**↑3.15	**45.48**↑5.08
♣ Qwen3-32B	70.37	54.81	29.17	8.00	40.59
♡ ReTool	70.00	61.11	38.89	6.67	44.17
♡ **EarlyTIR-32B**	**82.50**↑12.13	**69.58**↑14.77	**52.00**↑22.83	**12.80**↑4.80	**54.22**↑13.63
♣ Qwen2.5-7B-inst	7.33	5.33	3.67	1.20	4.38
♡ SimpleTIR-7B	**50.50**	30.90	**29.70**	-	-
♡ ZeroTIR-7B	39.60	25.00	22.50	-	-
♡ ReTool-7B	40.00	23.33	-	-	-
♡ **EarlyTIR-7B**	42.00↑34.67	**31.25**↑25.92	15.83↑12.16	0.75↓0.45	22.46↑18.08

5.2 Training Results

To evaluate the effectiveness of EarlyTIR, we compare its performance across models and tasks (math and code).

Mathematical Reasoning: As shown in Table 1, EarlyTIR outperforms most recent TIR baselines across all model scales and benchmarks. Notably, EarlyTIR-32B achieves a macro-average accuracy of 54.22%, a 10.05-point absolute improvement over the strong ReTool baseline. On the challenging AIME 2025, it reaches 69.58%, surpassing ReTool by 8.47%. The gap is particularly large for the Qwen3 series. By bounding invalid interactions, EarlyTIR keeps trajectories focused on high-quality steps and raises the capability ceiling of the 32B model above that of previous specialized TIR methods.

Code Generation: Similar trends are observed in coding tasks (Table 2). EarlyTIR demonstrates superior performance on HumanEval and LiveCodeBench compared to DeepCoder. In code-centric TIR, consecutive syntax errors or repeated execution timeouts often consume the context window with noise, leading to reasoning derailment. EarlyTIR's intervention prevents these error-retry loops from corrupting the agent's context. For instance, EarlyTIR-32B achieves 47.27% on LiveCodeBench, a +4.52% gain over the base model.

Table 2: Performance comparison on coding benchmarks. Our method consistently outperforms other RLVR-trained baselines. Notably, EarlyTIR achieves significant gains over the strong Qwen3-32B baseline. * denotes models trained on consistent data subsets for fair comparison.

Model	HumanEval	LiveCodeBench
Qwen3-8B	81.27	34.80
DeepCoder*	90.60	39.7
KodCode	89.31	33.46
MicroCoder	94.38	42.73
EarlyTIR*	93.4	42.80
Qwen3-32B	89.94	42.75
DeepCoder*	92.37	43.11
EarlyTIR*	**94.02** ↑4.08	**47.27** ↑4.52

Robustness Across Model Scales: The gains are not limited to a specific model size. EarlyTIR scales robustly, improving both 8B and 32B base models (e.g., Qwen3-8B and Qwen3-32B). This confirms that our intervention mechanism is model-agnostic and generalizes across different reasoning capabilities.

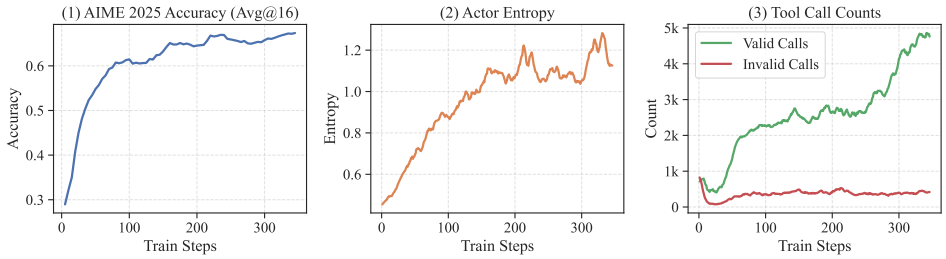


Figure 6: Training dynamics of EarlyTIR on Qwen3-32B. (1) Accuracy demonstrates stable growth while (2) policy entropy converges concurrently without decoupling. (3) The count of invalid tool calls is strictly suppressed.

5.3 Training Dynamics and Efficiency

To investigate how EarlyTIR shapes the learning process, we analyze the training curves of Qwen3-32B in Figure 6. We observe three effects. First, as shown in Figure 6(1), **Accuracy** rises steadily and reaches a high plateau without the late-stage degradation seen in standard RL. Second, **Actor Entropy** (2) converges together with accuracy. EarlyTIR prevents the policy from drifting into high-uncertainty regions by pruning non-informative trajectories. Most notably, **Tool Call Counts** (3) reveal a clear separation between valid and invalid interactions. While valid tool calls grow in tandem with reasoning capability, the invalid call count is strictly suppressed and remains near zero throughout training. This suggests that EarlyTIR filters out environmental noise, ensuring that the model's exploration budget is spent on high-quality reasoning steps rather than error-retry loops.

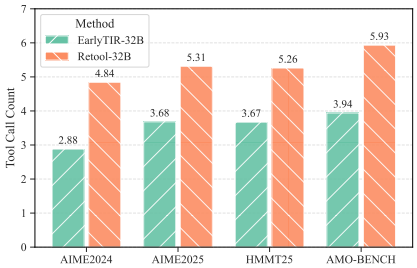


Figure 7: Comparison of average tool call counts per sample during evaluation. EarlyTIR-32B reaches solutions using fewer interactions than ReTool across all benchmarks.

Furthermore, EarlyTIR improves **inference efficiency**. As illustrated in Figure 7, EarlyTIR-32B consistently requires fewer tool calls than ReTool to reach a final answer on all benchmarks. Specifically, EarlyTIR reduces the average tool-use frequency by 30.2%–40.5%, with an average reduction of 33.7%. For instance, on the challenging AMO-Bench, the average call count drops from 5.93 to 3.94. This reduction means that EarlyTIR achieves better

reasoning performance with lower computational latency and environment overhead, making it more cost-effective for complex agentic tasks.

To test whether this efficiency gain is merely prompt-level steering, we repeat the AIME24 evaluation with the system prompt *"IMPORTANT: Rely on your own knowledge first. Use tools ONLY if absolutely necessary."* As shown in Table 3, the prompt reduces tool calls for both methods, but it hurts ReTool accuracy by 4.73 points while EarlyTIR drops by only 1.12 points. This indicates that EarlyTIR's efficiency advantage is not a byproduct of prompting the model to use fewer tools; the policy has learned to use tools more effectively.

Table 3: Effect of an explicit tool-discouraging prompt on AIME24. Each cell reports accuracy (%) / average tool calls.

Method	**w/o prompt**	**w/ prompt**
ReTool-32B	70.00 / 4.84	65.27 / 3.76
EarlyTIR-32B	82.50 / 2.88	81.38 / 2.44

5.4 Ablation Studies

We conduct ablation studies to isolate the impact of core hyperparameters in EarlyTIR using the Qwen3-32B model. Results are summarized in Table 4.

Impact of Tolerance Threshold (τ): Table 4 shows a non-monotonic relationship between τ and accuracy. We find that $\tau = 2$ yields the best overall performance (82.50 on AIME24 and 69.58 on AIME25) and adopt it as the default. A larger tolerance does not bring consistent gains: $\tau = 3$ and $\tau = 5$ degrade accuracy on both benchmarks, while $\tau = 4$ slightly exceeds $\tau = 2$ on AIME25 (70.00 vs. 69.58) but remains 3.61 points lower on AIME24. This suggests that a small tolerance window is sufficient for learning error recovery, whereas larger values admit too many non-informative interactions and dilute the training signal. The slight advantage of $\tau = 4$ on the harder AIME25 subset nevertheless indicates that the optimal tolerance may scale with task difficulty, which motivates the adaptive thresholds discussed in Appendix A. These results yield a simple practical recipe: use $\tau = 2$ for simpler tasks, where aggressive pruning is safe, and $\tau = 4$ for harder task distributions, where a larger recovery buffer helps. We therefore treat $\tau \in \{2, 4\}$ as the recommended range and leave adaptive selection to future work.

Truncation Penalty Reward (R_t): The intensity of the negative reinforcement signal is critical. A mild penalty ($R_t = -0.2$) yields the best results by providing a balanced guidance signal. As the penalty increases to -1.0, performance degrades toward the base model level. This indicates that an overly severe penalty discourages effective tool use and prevents the model from leveraging external feedback.

Necessity of Early Truncation: To separate the effect of the penalty from that of truncation, we introduce *No-Truncat*, which applies the same $R_t = -0.2$ penalty but lets the trajectory continue to the maximum turn limit. No-Truncat improves AIME24 by only 1.24 points and slightly degrades AIME25 (60.87 vs. 61.11), whereas EarlyTIR with $\tau = 2$ improves them by 12.50 and 8.47 points, respectively. This confirms that penalizing invalid calls without halting the rollout cannot prevent the model from wasting its context on error-retry loops; early truncation is the key ingredient.

Table 4: Ablation of EarlyTIR on tolerance threshold (τ) and penalty reward (R_t) on Qwen3-32B performance.

Setting	**AIME2024**	**AIME2025**
ReTool (base)	70.00	61.11
Tolerance Threshold τ		
w/. $\tau = 2$	**82.50** ↑12.50	69.58 ↑8.47
w/. $\tau = 3$	72.92 ↑2.92	60.83 ↓0.28
w/. $\tau = 4$	78.89 ↑8.89	**70.00** ↑8.89
w/. $\tau = 5$	80.00 ↑10.00	64.44 ↑3.33
Truncation vs. Penalty		
No-Truncat	71.24 ↑1.24	60.87 ↓0.24
Penalty Reward R_t		
w/.$R_t = -0.2$	**82.50** ↑12.50	**69.58** ↑8.47
w/. $R_t = -0.5$	71.11 ↑1.11	61.11 ↑0.00
w/. $R_t = -1.0$	70.00 ↑0.00	65.56 ↑4.45

6 Related Work

Tool-Integrated Reasoning (TIR). TIR empowers Large Language Models (LLMs) to overcome inherent computational constraints by offloading symbolic computation to external tools such as code interpreters and APIs. Early paradigms primarily relied on prompt engineering [3, 31] and Supervised Fine-Tuning (SFT) [6, 28, 23]. However, SFT-driven models often suffer from behavioral cloning, where they merely imitate demonstrations without developing the robust, iterative error-correction capabilities required for complex tasks.

Recent advances in Agentic RL have shifted the focus toward reinforcement learning with verifiable rewards (RLVR) [7, 32, 34]. By treating tool execution as an explicit action space and optimizing policies via outcome-driven signals (e.g., unit test results), these methods allow models to transcend static supervision. Frameworks such as ReTool [5], ToRL [10], and SimpleTIR [27] show that closing the loop between execution and learning substantially improves success rates on complex problems compared with text-only reasoning.

Entropy Dynamics in Reinforcement Learning. A growing body of research suggests that the performance ceiling of LLMs in RL is inextricably linked to policy entropy [4, 24, 18, 30, 36]. For example, Cui et al. [4] established a mathematical model (the R-H relationship) showing that entropy is a prerequisite for exploration. Premature entropy exhaustion often leads to early collapse in long-horizon reasoning [30]. Furthermore, Wang et al. [24] revealed that RL efficiency is driven by a high-entropy minority of tokens, i.e., critical logical pivots at which the model's stochasticity is highest.

Despite these insights, existing studies on entropy mechanisms are largely confined to text-only reasoning. The introduction of tool calls alters the entropy landscape, as external feedback provides new information that can either resolve or exacerbate policy uncertainty. Research on entropy dynamics in TIR remains scarce [11]. A notable exception is the work by Li et al. [11], which reveals that high-quality tool calls reduce entropy in multi-turn TIR.

Trajectory Filtering and Concurrent Work. SimpleTIR performs offline, trajectory-level filtering: complete rollouts are scored and low-quality samples are discarded after generation. In contrast, EarlyTIR operates online during rollout, uses the real-time invalid-call count as the signal, and halts hallucination loops mid-generation. The truncated trajectory is retained as a negative sample rather than discarded. This distinction matters: preserving the context window leaves room for subsequent high-quality steps, and it enables our entropy-decoupling analysis, which SimpleTIR does not address. Concurrent work by Wei et al. [25] uses predictive entropy to guide branching at inference time in large tool spaces and reports an error-entropy correlation. Our focus is complementary: we identify training-time entropy–performance decoupling across model families and intervene in RL rollouts to suppress invalid-call accumulation, rather than expanding search at test time.

7 Conclusions

This paper identifies an entropy–performance decoupling phenomenon in agentic RL for TIR, revealing that invalid tool interactions drive ineffective policy uncertainty during late-stage training. To resolve this, we propose EarlyTIR, a strategic truncation mechanism that prunes non-informative trajectories while preserving the model's ability to learn error-recovery behaviors. Extensive evaluations demonstrate that EarlyTIR consistently achieves good accuracy across diverse benchmarks while significantly enhancing inference efficiency by reducing invalid tool interactions.

Acknowledgments and Disclosure of Funding

The research in this article is supported by the New Generation Artificial Intelligence of China (2024YFE0203700), National Natural Science Foundation of China under Grants 62576124.

References

[1] Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma, Xuanlin Wang, Ziwen Wang, et al. Amo-bench: Large language models still struggle in

high school math competitions. *arXiv preprint arXiv:2510.26768*, 2025.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2022.

[4] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

[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. *arXiv preprint arXiv:2504.11536*, 2025.

[6] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[8] Harvard-MIT Mathematics Tournament. Harvard-mit mathematics tournament (hmmt) february 2025. `https://www.hmmt.org/www/archive/282`, 2025. Accessed: 2025-02-15.

[9] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[10] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

[11] Zeping Li, Hongru Wang, Yiwen Zhao, Guanhua Chen, Yixia Li, Keyang Chen, Yixin Cao, Guangnan Ye, Hongfeng Chai, and Zhenfei Yin. Rethinking the role of entropy in optimizing tool-use behaviors for large language model agents. *arXiv preprint arXiv:2602.02050*, 2026.

[12] Zongqian Li, Tengchao Lv, Shaohan Huang, Yixuan Su, Qinzheng Sun, Qiufeng Yin, Ying Xin, Scarlett Li, Lei Cui, Nigel Collier, et al. Scaling data difficulty: Improving coding models via reinforcement learning on fresh and challenging problems. *arXiv preprint arXiv:2603.07779*, 2026.

[13] Heng Lin and Zhongwen Xu. Understanding tool-integrated reasoning. *arXiv preprint arXiv:2508.19201*, 2025.

[14] Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025. Notion Blog.

[15] Xinji Mai, Haotian Xu, Zhong-Zhi Li, Weinong Wang, Jian Hu, Yingying Zhang, Wenqiang Zhang, et al. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving. *arXiv preprint arXiv:2505.07773*, 2025.

[16] Xinji Mai, Haotian Xu, Weinong Wang, Yingying Zhang, Wenqiang Zhang, et al. Agentic rl scaling law: Spontaneous code execution for mathematical problem solving. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[17] Mathematical Association of America. American invitational mathematics examination (aime), 2024. URL `https://maa.org/news/aime-thresholds-are-available/`.

[18] Aleksei Petrenko, Ben Lipkin, Kevin Chen, Erik Wijmans, Marco Cusumano-Towner, Raja Giryes, and Philipp Krähenbühl. Entropy-preserving reinforcement learning. *arXiv preprint arXiv:2603.11682*, 2026.

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

[20] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[21] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[22] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[23] Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*, 2023.

[24] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

[25] Rongzhe Wei, Ge Shi, Min Cheng, Na Zhang, Pan Li, Sarthak Ghosh, Vaibhav Gorde, and Leman Akoglu. Long-horizon plan execution in large tool spaces through entropy-guided branching. *arXiv preprint arXiv:2604.12126*, 2026.

[26] Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6980–7008, 2025.

[27] 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. *arXiv preprint arXiv:2509.02479*, 2025.

[28] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

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

[30] Kai Yang, Xin Xu, Yangkun Chen, Weijie Liu, Jiafei Lyu, Zichuan Lin, Deheng Ye, and Saiyong Yang. Entropic: Towards stable long-term training of llms via entropy stabilization with proportional-integral control. *arXiv preprint arXiv:2511.15248*, 2025.

[31] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544, 2024.

[32] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[33] 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. *arXiv preprint arXiv:2503.18892*, 2025.

[34] Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, et al. Tool zero: Training tool-augmented llms via pure rl from scratch. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9135–9147, 2025.

[35] Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025.

[36] Jinghan Zhang, Xiting Wang, Fengran Mo, Yeyang Zhou, Wanfu Gao, and Kunpeng Liu. Entropy-based exploration conduction for multi-step reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3895–3906, 2025.

A Limitations and Discussion

EarlyTIR improves both reasoning accuracy and training stability, but several limitations remain. We summarize them below and outline directions for future work.

Heuristic-based Truncation Thresholds. EarlyTIR relies on a pre-defined, fixed tolerance threshold τ. Our ablation shows that the optimal τ varies with model scale and task complexity. A fixed value may be too restrictive for smaller models that need more exploration, or too lenient for larger, more efficient models. Our results suggest a practical starting point: $\tau = 2$ for simpler tasks, where aggressive pruning is safe, and $\tau = 4$ for harder task distributions, where a larger recovery buffer helps. Future work could explore adaptive truncation that adjusts τ online based on policy confidence or the cumulative information gain of a trajectory.

Formal Theoretical Framework. Our Spearman's rank correlation ($\rho = 0.93$) provides strong empirical evidence for the coupling between invalid tool calls and ineffective entropy growth, but a rigorous mathematical derivation is still missing. Formally modeling the relationship between environmental information gain and policy uncertainty in Tool-Integrated Reasoning (TIR) remains an open problem. A theoretical bound on when environment noise begins to inject "ineffective entropy" into the actor could lead to more principled RL objectives.

Generalization to Stochastic and Open-ended Domains. This study focuses on verifiable tasks (mathematics and coding) with deterministic environments and binary rewards. The efficacy of EarlyTIR in more stochastic or subjective tool-use scenarios remains unexplored. Examples include web browsing, open-domain question answering, and interaction with noisy APIs, where rewards are sparse and execution feedback is ambiguous.

B Contributions

Our contributions are summarized as follows:

- Discovery of Entropy Decoupling: We provide the first systematic evidence of entropy–performance decoupling in Agentic RL for TIR, identifying it as a key source of training stagnation.
- Mechanism Analysis: We quantify the link between invalid tool trajectories and ineffective entropy, characterizing the "hallucination loops" that bottleneck long-horizon reasoning.
- EarlyTIR Framework: We introduce a novel early-intervention mechanism that strategically prunes non-informative exploration while preserving error-recovery learning. It shows performance gains in both accuracy and inference efficiency.

C Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

D LLM USAGE

During the preparation of this work, we used LLM to aid in polishing the manuscript and improving language. All final content was reviewed and revised by the authors to ensure its accuracy and originality. The core ideas, methods, and conclusions of the paper are solely the work of the authors.

E Detailed Experimental Results

E.1 Maximum Tool-use Turns Study

Maximum Tool-use Turns (8 vs. 16): Figures 8 and 9 compare standard RL training on Qwen3-8B with an 8-turn and a 16-turn cap. The 16-turn run ends at a lower and less stable AIME25 accuracy plateau (about 55% vs. 62%) while accumulating more valid and invalid tool calls and exhibiting continuously rising entropy. This indicates that over-extended horizons increase variance and redundant interaction, so we cap training at 8 turns. This comparison serves as the trajectory-length control for our correlational analysis: all EarlyTIR experiments use the same 8-turn cap as the main results, and the 16-turn run characterizes the horizon effect under the shared training configuration.



Figure 8: Standard RL on Qwen3-8B with a maximum of 8 tool-use turns.



Figure 9: Standard RL on Qwen3-8B with a maximum of 16 tool-use turns.

E.2 Pass@k Evaluation

To check whether EarlyTIR narrows the solution distribution, we sample each problem 8 times from the trained Qwen3-32B checkpoint and compute Pass@k. The corresponding base-model Pass@1 is 70.37 on AIME24 and 54.81 on AIME25. As shown in Table 5, Pass@k increases monotonically with k on both benchmarks, indicating that diverse valid solution paths remain reachable. This analysis is supplementary: it uses a single checkpoint and does not include Pass@k results for the baselines, so we do not use it to support the main accuracy claims.

Table 5: Pass@k on AIME for EarlyTIR-32B. Each problem is sampled 8 times from a single trained checkpoint.

Benchmark	Pass@1	Pass@3	Pass@8
AIME24	82.50	88.81	93.33
AIME25	69.58	78.75	83.33

E.3 Invalid Tool-call Detector Validation

Detection rules. Table 6 summarizes the four deterministic signals used by the rule-based detector. These signals are recorded by the execution sandbox rather than inferred by an LLM.

Table 6: Invalid tool-call detection rules.

Failure mode	Sandbox signal	Rule
Execution error	stderr / exit code	non-empty stderr or non-zero exit code
Timeout	executor timer	timer interrupt raised
Empty return	stdout	missing, `None`, or empty stdout
Redundant command	call history	current call identical to the previous call

Audit. We sampled 50 held-out trajectories and manually compared every detected invalid call against the system logs. For the four error types above, the detector achieved 100% precision: every call it flagged was confirmed invalid. The four categories account for more than 85% of the non-informative interactions observed in our pre-study. The audit measures precision rather than recall; in particular, exact-match redundancy can flag a valid re-execution of the same command after the environment state has changed.

Uncovered Invalid Interactions. Our detector covers only invalid interactions that leave deterministic signals in the sandbox. Other forms of invalid interaction remain outside its scope. Examples include tool calls that execute successfully but are irrelevant to the current sub-goal, repeated attempts that differ only superficially, and non-empty outputs that provide no useful information for the problem. Detecting these cases typically requires an additional LLM-based judge. Running such a judge on every rollout would consume substantial computation and latency, and would therefore add a large training overhead. We therefore restrict EarlyTIR to rule-based signals, which are cheap and reliable, and leave broader coverage to future work.

F Detailed RLVR Algorithms

F.1 RLVR Algorithms: GRPO and DAPO

Group Relative Policy Optimization [20] efficiently optimizes the model's reasoning capabilities without the memory overhead of a value network. Unlike Proximal Policy Optimization [19], which relies on a separate value function to estimate the baseline, GRPO leverages the distribution of rewards within a group of sampled outputs to compute the advantage.

Group Sampling and Advantage Estimation. For each query q, GRPO samples a group of G outputs $\{o_1, o_2, \ldots, o_G\}$ from the current policy π_θ. Let r_i be the reward for the i-th output. The advantage A_i is computed by normalizing the reward against the group statistics:

$$A_i = \frac{r_i - \bar{r}}{\sigma_r + \epsilon} \tag{1}$$

where $\bar{r}$ and σ_r are the mean and standard deviation of the rewards within the group, respectively. This group-based baseline serves as a dynamic reference, reducing variance without requiring a learned value function.

Objective Function. The optimization objective maximizes the surrogate loss with a KL-divergence penalty to maintain stability relative to the reference model π_{ref}:

$$\mathcal{J}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{o_i\}\sim\pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^{G} \left(\mathcal{L}_{\text{clip}}(r_t(\theta), A_i) - \beta \mathbb{D}_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) \right) \right] \tag{2}$$

Here, $r_t(\theta) = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$ denotes the probability ratio, and $\mathcal{L}_{\text{clip}}$ is the standard PPO clipping operator:

$$\mathcal{L}_{\text{clip}}(r_t, A_i) = \min\left(r_t A_i, \text{clip}(r_t, 1-\epsilon, 1+\epsilon) A_i\right) \tag{3}$$

By eliminating the critic model, GRPO significantly reduces computational costs, enabling efficient training of large-scale reasoning models.

Decoupled Clip and Dynamic sAmpling Policy Optimization (DAPO). While GRPO offers efficiency, it often suffers from entropy collapse and training instability when dealing with complex reasoning and tool-call sequences. To address this, we employ DAPO as the RLVR algorithm in this paper. It introduces four key techniques to stabilize RL in long-CoT scenarios: (1) Clip-Higher, which promotes the diversity of the system and avoids entropy collapse; (2) Dynamic Sampling, which improves training efficiency and stability; (3) Token-Level Policy Gradient Loss, which is critical in long-CoT RL scenarios; (4) Overlong Reward Shaping, which reduces reward noise and stabilizes training.

Overall, DAPO samples a group of outputs $\{o_i\}_{i=1}^{G}$ for each question q paired with the answer a, and optimizes the policy via the following objective:

$$\begin{aligned}
\mathcal{J}_{\text{DAPO}}(\theta) =& \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \\
& \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G} \sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})\hat{A}_{i,t} \right) \right] \\
& \text{s.t.} \quad 0 < |\{o_i \mid \text{is_equivalent}(a, o_i)\}| < G,
\end{aligned} \tag{4}$$

where

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, \quad \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}. \tag{5}$$

G Setting Details

RLVR Implementation: We implemented DAPO using the `verl` framework. Code execution was hosted in a sandboxed Python environment for security and reproducibility. During the rollout phase, we used vLLM for high-throughput generation. To support the ultra-long context window in multi-turn interactions, our training infrastructure leveraged Fully Sharded Data Parallel (FSDP) with CPU offloading and gradient checkpointing. We also applied Ulysses Sequence Parallelism (SP=4) for the actor model and Tensor Parallelism (TP=4) for rollout generation. For 7/8B-scale models, each RL session was completed within 72 hours on a cluster of 64 Ascend 910B NPUs (8 nodes). For 32B-scale models, each RL session was completed within 96 hours on a cluster of 64 Ascend 910B NPUs (8 nodes).

At evaluation time, we use the fully converged checkpoint of each trained model. To ensure reliable and reproducible assessment of tool-use capabilities, inference runs in the same sandboxed Python environment used during RLVR training. Inference is accelerated by vLLM with Tensor Parallelism to meet the model's memory demands on Ascend 910B NPUs.

Table 7: Hyperparameters for DAPO training.

Hyperparameter	Value
Data & Rollout	
Rollout Batch Size (prompts)	512
PPO Mini-batch Size (prompts)	128
Max Prompt/Response Length	2k / 16k
Sampling Temperature	1.0
Top-p	0.6
Rollout Group Size G	16
Max Tool-call Turns	8
RL Optimization	
Learning Rate	1×10^{-6}
Optimizer	AdamW
Clip Ratio (L/H/C)	0.2/0.28/10
KL Reference Loss	0.0
Weight Decay	0.01
EarlyTIR	
Invalid-call Tolerance τ	2
Truncation Penalty R_t	-0.2

Table 8: Hyperparameters setup for inference and evaluation.

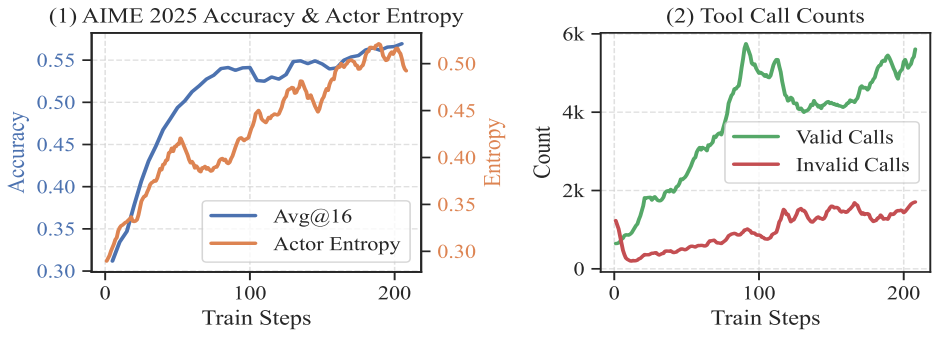
Configuration	Value
Decoding Strategy	
Temperature	1.0
Top-p	0.6
Max Prompt Length	2,048
Max Generation Length	16,384
Hardware Acceleration	
Inference Engine	vLLM
TP	4/8
Hardware	Ascend 910B

H Supplementary Analyses across Models

In this section, we provide supplementary analyses of entropy dynamics and tool-use behavior across models with varying parameter scales and architectures. These results complement the findings in the main text (specifically Figures 2 and 4).

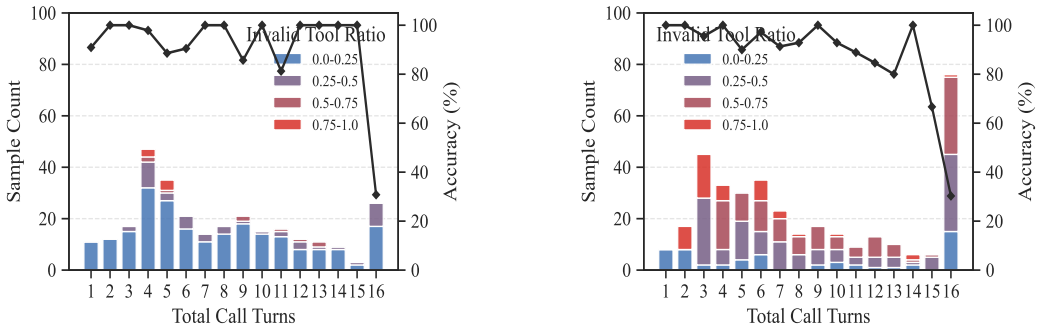
We first present the overall entropy dynamics on the Math task across multiple foundation models. Figure 10 details the behavior of Qwen3-8B under the EarlyTIR setting. To test whether the accuracy–invalidity relationship holds beyond the Qwen family, Figure 11 extends the accuracy-versus-turn analysis to two API-based models, DeepSeek-V3.2 and Kimi-2.5. Both exhibit the same pattern: as trajectories lengthen and the invalid-call ratio increases, accuracy degrades.

We further examine how model confidence evolves over multi-step interactions. Figure 12 shows turn-level average entropy during RL training for the Qwen3 family (8B and 32B). A consistent pattern emerges across all evaluated parameter scales: as training progresses, the average entropy of high-round samples gradually increases and eventually surpasses that of low-round samples.



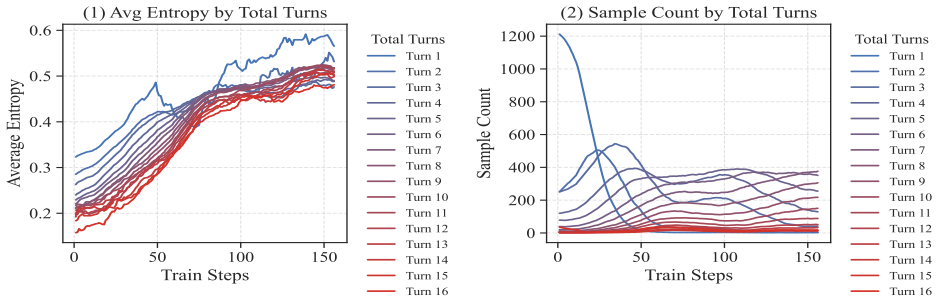
(a) Qwen3-8B (EarlyTIR)

Figure 10: Supplementary results of entropy dynamics on the Math task. As a complement to Figure 2, we present the results for EarlyTIR-8B.

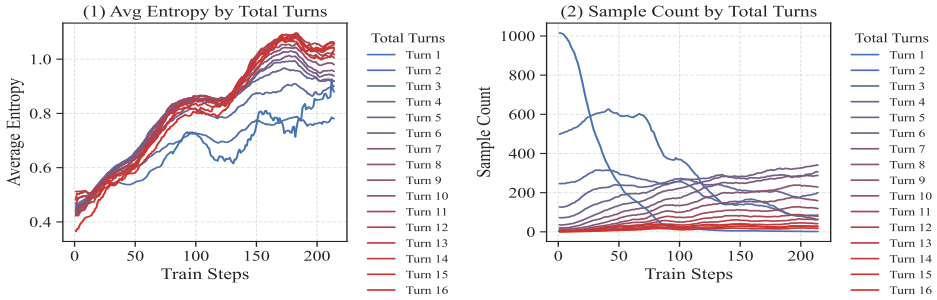


(a) Kimi-2.5 (b) DeepSeek-V3.2

Figure 11: Accuracy versus tool-call turns for API-based models on the Math task. Stacked bars show the sample count in each turn bin, color-coded by invalid-tool ratio; the black line shows accuracy. As a complement to Figure 4, longer trajectories with a high invalid-call ratio coincide with lower accuracy for both DeepSeek-V3.2 and Kimi-2.5.



(a) Qwen3-8B



(b) Qwen3-32B

Figure 12: Turn-level average entropy dynamics during RL training. As RL training progresses, the average entropy of high-round samples gradually surpasses that of low-round samples; this pattern is consistent across the evaluated parameter scales in the Qwen3 series.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[N/A]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims regarding the entropy-performance decoupling phenomenon and the effectiveness of EarlyTIR, which are supported by the subsequent empirical evaluations.

   Guidelines:

   - The answer NA 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 NA 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 have dedicated Section A to discuss the limitations.

Guidelines:

- The answer NA 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: [N/A]

   Justification: The paper is primarily empirical and does not introduce new theoretical proofs or theorems.

   Guidelines:

   - The answer NA 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: We provide comprehensive details on the experimental setup, datasets, and hyperparameters in Sections 3.1, 5.1, and Appendices F and G.

   Guidelines:

- The answer NA 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: An anonymized version of our code and instructions are included in the supplemental material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All necessary details, including batch sizes, learning rates, sampling temperatures, and optimization parameters, are clearly specified in Section 5.1 and Tables 7 and 8 in Appendix G.

   Guidelines:

   - The answer NA 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: Experimental results in Table 1 are reported as the mean over 3 independent runs with different random seeds.

   Guidelines:

   - The answer NA 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: Appendix G discloses GPU resources, inference engines, and the sandbox environments used.

Guidelines:

- The answer NA 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 reviewed the NeurIPS Code of Ethics and confirm that our study adheres to all ethical guidelines.

   Guidelines:

   - The answer NA 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: We include an Impact Statement in Appendix C acknowledging the broader societal context of our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: As stated in Section 3.1 and Appendix G, all tool and code executions during our training and evaluation were strictly hosted in a local sandboxed Python environment to ensure security.

Guidelines:

- The answer NA 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 properly cite all utilized open-source foundation models (e.g., Qwen, DeepSeek) and datasets (e.g., DeepCoder, AIME) in the main text.

Guidelines:

- The answer NA 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 paper proposes a new algorithmic intervention mechanism (EarlyTIR) during training, but does not introduce or release novel standalone datasets or base models.

Guidelines:

- The answer NA 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 research relies solely on automated programmatic benchmarks and does not involve any crowdsourcing or human subjects.

    Guidelines:

    - The answer NA 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 research relies solely on automated programmatic benchmarks and does not involve any crowdsourcing or human subjects.

    Guidelines:

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs were only used for language editing and formatting (D), which does not impact the core methodology.

    Guidelines:

    - The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.