

# E<sup>3</sup>-TIR: Enhanced Experience Exploitation for Tool-Integrated Reasoning

Weiyang Guo<sup>1</sup> Zesheng Shi<sup>1</sup> Liye Zhao<sup>2</sup> Jiayuan Ma<sup>1</sup>  
Zeen Zhu<sup>1</sup> Junxian He<sup>3</sup> Min Zhang<sup>1</sup> Jing Li<sup>1</sup>✉

<sup>1</sup>Harbin Institute of Technology, Shenzhen, China

<sup>2</sup>Huawei Technologies Co., Ltd.

<sup>3</sup>Hong Kong University of Science and Technology

weiyanguo2024@gmail.com jingli.phd@hotmail.com

## Abstract

While Large Language Models (LLMs) have demonstrated significant potential in Tool-Integrated Reasoning (TIR), existing training paradigms face significant limitations: Zero-RL suffers from inefficient exploration and mode degradation due to a lack of prior guidance, while SFT-then-RL is limited by high data costs and capability plateaus caused by low-entropy collapse. To address these challenges, we propose E<sup>3</sup>-TIR (Enhanced Experience Exploitation), a warm-up paradigm for the early stages of agent training. Specifically, we formulate training as the dynamic integration of three experience types: Expert Prefixes, Expert Guided, and Self-Exploration. By executing diverse branching exploration around expert “anchors” and employing a mix policy optimization mechanism, we effectively mitigate distribution shifts and resolve optimization conflicts arising from shared prefixes. Our method dynamically adapts the model’s knowledge boundaries, effectively balancing exploration diversity with training efficiency. Experimental results demonstrate that E<sup>3</sup>-TIR achieves a 6% performance improvement over traditional paradigms on tool-use tasks, while requiring less than 10% of the synthetic data. Furthermore, in terms of ROI—a comprehensive metric integrating performance, data cost, and training efficiency—we achieve a 1.46× gain compared to baselines. Code is available at <https://github.com/yuki-younai/E3-TIR>.

## 1 Introduction

Large Language Models (LLMs) have demonstrated exceptional potential in complex reasoning and tool-assisted decision-making, which has driven the realization of advanced applications such as question answering, planning, and interactive agents (Zhang et al., 2025; Zhao et al., 2025). To

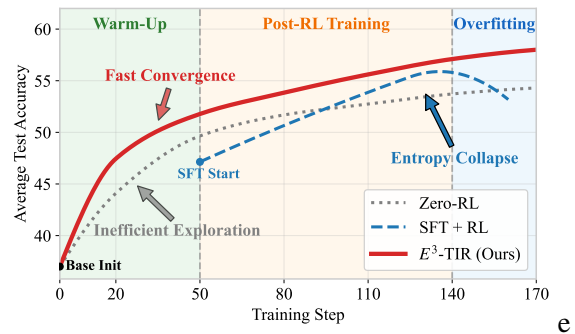


Figure 1: (a) Zero-RL: Inefficient exploration and slow convergence. (b) SFT+RL: Vulnerable to overfitting and performance collapse after a high start. (c) E<sup>3</sup>-TIR: Enables rapid adaptation via expert guidance with stable, continuous growth.

overcome the limitations of purely internal reasoning, LLMs are often deployed as Tool-Integrated Reasoning (TIR) agents (Dong et al., 2025a). A TIR agent performs reasoning and action through a multi-step, interleaved process that invokes external tools such as code interpreters (Xue et al., 2025) and search engines (Wang et al., 2025b).

Recently, reinforcement Learning (RL) has shown significant promise in unlocking frontier LLM capabilities (Shao et al., 2024; YANG et al., 2026), methods such as Search-R1 (Jin et al., 2025a) and ToRL (Li et al., 2025b) train directly on the foundation model, utilizing self-exploration to acquire tool-use capabilities, but this compromises training efficiency. Conversely, approaches like Tool-Star (Dong et al., 2025b) and Tool-Light (Chen et al., 2025c) necessitate an initial stage of supervised fine-tuning with synthetic data before engaging in reinforcement learning. Reliance on synthetic data is not only computationally expensive but also introduces training redundancy.

Our empirical analysis further reveals the deep-seated limitations of existing paradigms. Zero-RL, lacking prior guidance, exhibits inefficient exploration and mode degradation: the model requires

✉ Corresponding author.

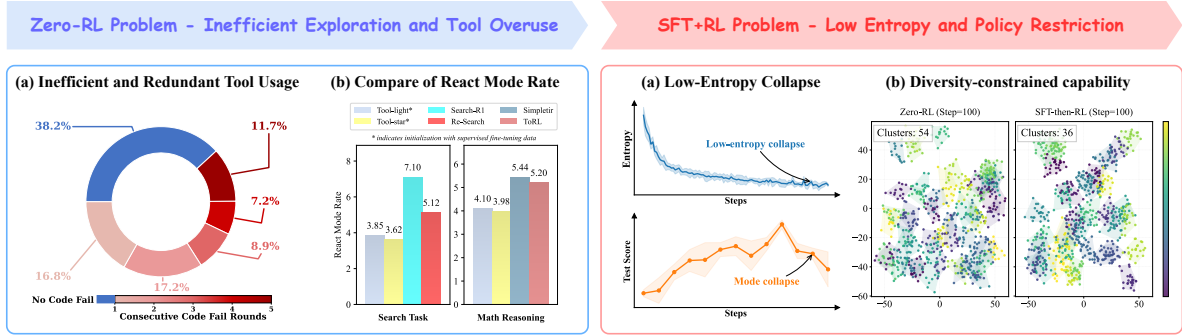


Figure 2: Statistical analysis of the limitations of current training paradigms.

extensive trial-and-error to generate valid code at an early stage (Fig. 2 left,a) and often converges to a “react mode” (Yu et al., 2025) that over-relies on tools at the expense of reasoning (Fig. 2 left, b). Conversely, SFT-then-RL, while benefiting from training, leads to capability rigidity. This necessitates an adaptation period to overcome distribution shifts (Fig. 2 right,a) and frequently triggers low-entropy collapse in the later stages of training (Fig. 2 right,b), which severely restricts sample diversity and impedes further performance gains.

To address these challenges, we propose E<sup>3</sup>-TIR, a framework designed to balance exploration diversity with efficiency. In the sampling stage, we initiate branching sampling from high-entropy “anchors” within expert trajectories to explore diverse reasoning paths. Subsequently, we dynamically filter the mixed trajectories based on gradient validity and relative performance. For mix policy optimization, we employ hybrid advantage estimation to capture preference variances among branches sharing the same prefixes. Meanwhile, advantage blocking resolves gradient conflicts on shared prefixes, ensuring training stability.

The main contributions of this study are as follows:

- Through empirical analysis, we uncover the “react mode” inherent in Zero-RL tool call, alongside the phenomena of low-entropy collapse and capability rigidity that occur in the later stages of SFT+RL.
- We propose E<sup>3</sup>-TIR, which provides dynamic experience sources for an agent’s early training by integrating three distinct types of experience: expert prefixes, expert guidance, and self-exploration. This approach effectively balances the diversity and efficiency of early-stage exploration.
- E<sup>3</sup>-TIR outperforms SFT and RL baselines

using <10% of the synthetic data. With a minimal warm-up, it achieves a 1.46× gain in ROI—a composite metric of data, training, and performance efficiency.

## 2 Preliminaries

### 2.1 Multi-Turn Agent Framework

We consider an LLM agent performing a task  $x \in D$  through iterative interactions. At each step  $t$ , the policy  $\pi_\theta(\cdot|\mathcal{H}_{<t})$  generates a thought  $\tau_t$  and an action  $a_t$ , eliciting an observation  $o_t$  from the environment. This yields a trajectory  $\mathcal{H} = \{(\tau_t, a_t, o_t)\}_{t=1}^T$ . As the trajectory length grows linearly with the number of steps, optimizing  $\pi_\theta$  for complex tasks becomes increasingly challenging due to the accumulation of history.

### 2.2 Agentic Reinforcement Learning

Agentic RL typically adopts policy-gradient methods to optimize the agent policy  $\pi_\theta$ . We formulate the agentic RL training objective as:

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, \mathcal{H} \sim \pi_\theta(\cdot|x)} [r_\phi(x, \mathcal{H})] - \beta \mathbb{D}_{\text{KL}} [\pi_\theta(\cdot) \| \pi_{\text{ref}}(\cdot)] \quad (1)$$

where  $\pi_\theta$  represents the policy LLM,  $\pi_{\text{ref}}$  is the reference LLM,  $r_\phi$  and  $\mathbb{D}_{\text{KL}}$  denotes the reward function and KL divergence respectively. To ensure proper credit assignment, we implement token masking ( $o_t$ ). The loss is computed exclusively on the agent’s outputs ( $\tau_t$  and  $a_t$ ), thereby excluding environmental feedback from gradient updates.

## 3 Related Work

**Tool Integrated Reasoning.** Tool-Integrated Reasoning (TIR) (Lin and Xu, 2025; Singh et al., 2025) empowers LLMs to overcome the inherent limitations of pure in-context reasoning by leveraging external utilities such as code interpreters and search engines. For question-answer

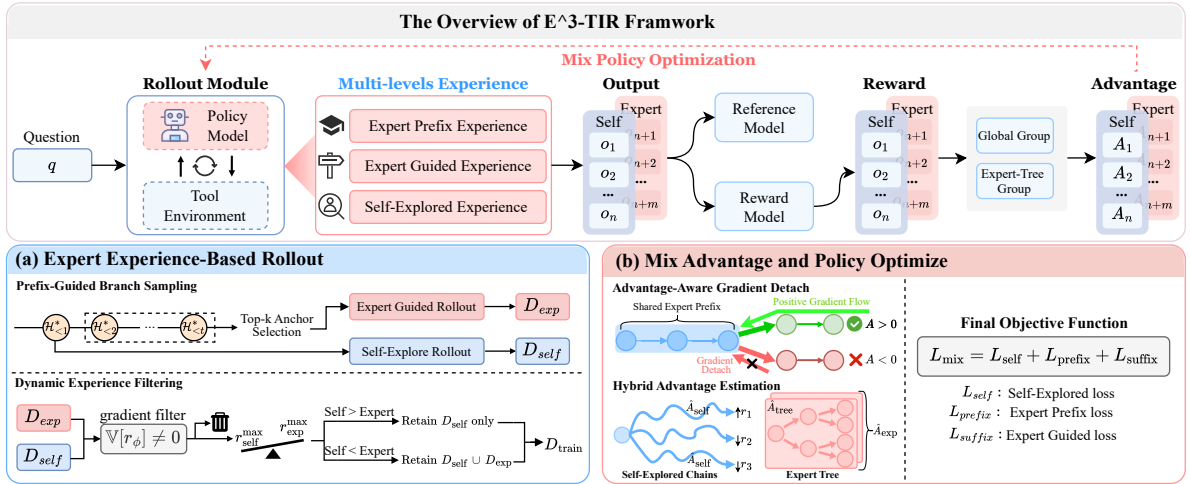


Figure 3: Illustration of our  $E^3$ -TIR framework. (a) Branching exploration from expert anchors and dynamic experience filtering. (b) Hybrid advantage estimation and advantage-aware gradient detachment.

(QA) tasks, efforts such as Search-R1 (Jin et al., 2025a), R1-Search (Song et al., 2025), and Research (Chen et al., 2025a) primarily focus on utilizing search tools. Conversely, in mathematical tasks, TIR largely relies on agent-generated Python code to enhance reasoning accuracy and efficiency. Current methodologies for training LLMs in tool invocation present a distinct set of challenges. One path involves Zero-shot RL based online exploration, where models like ToRL (Li et al., 2025b), and SimpleTIR (Xue et al., 2025) learn autonomous tool calls. However, this often leads to inefficient exploration and redundant tool invocation. The second path utilizes Supervised Fine-Tuning (SFT) for expert knowledge injection, with works like ReTool (Feng et al., 2025), Toolstar (Dong et al., 2025b), focusing on synthesizing high-quality agent data to inject tool-use priors before RL training.

**Agent Reinforcement Learning.** Reinforcement Learning (RL) plays a pivotal role in enabling LLMs agents to adapt effectively to dynamic and open-ended environments (Li et al., 2026). Recent efforts have primarily focused on addressing the challenges of sparse rewards (Xue et al., 2026a) and complex decision-making in multi-turn interactions (Singh et al., 2025). RAGEN (Wang et al., 2025c) integrates the reasoning state with the environmental interaction into turn-level responses and trains the agent using trajectory-level RL. To further alleviate the issue of reward sparsity, IGPO (Wang et al., 2025a) introduce more fine-grained rewards, such as leveraging the agent’s self-confidence. To overcome the difficulty of captur-

ing complex multi-turn behaviors with trajectory-level RL, Tree-GRPO (Ding and Ye, 2025) employs a Grouped Relative Policy Optimization based on tree search, allowing the model to effectively learn intermediate decisions. Furthermore, ARPO (Dong et al., 2025c) secures more diverse reasoning paths through high-entropy branch sampling, and AEPO (Dong et al., 2025a) advances this direction by proposing a policy optimization method based on entropy balancing.

## 4 Methodology

In our  $E^3$ -TIR framework, we divide the overall training flow into two main parts:

1) **Expert Experience-Based Rollout.** First, we utilize the current policy to conduct self-exploration and expert-guided exploration, respectively, performing branch sampling from the expert prefixes to form the experience pools  $\mathcal{D}_{exp}$  and  $\mathcal{D}_{self}$  (Section 4.1.1). Subsequently, we filter the experiences pools based on gradient validity and performance advantage to construct the final training set  $\mathcal{D}_{train}$  (Section 4.1.2).

2) **Mix Advantage and Policy Optimization.** We categorize advantages into global advantage and expert-tree internal advantage to determine relative trajectory preference, processing different experiences separately (Section 4.2.1). To update the policy using mixed experiences, we introduce an off-policy shape and resolve gradient conflicts in shared prefixes via advantage-aware gradient blocking (Section 4.2.2).

## 4.1 Expert Experience-Based Rollout

### 4.1.1 Prefix-Guided Branch Sampling

Unlike self-exploration methods (Dong et al., 2025c,a), we propose expert-guided branch sampling, using expert prefixes as anchors to reduce search redundancy and ensure reasoning correctness (Theoretical Proof provided in Appendix E).

Empirical analysis confirms this advantage: branching from expert prefixes yields significantly higher policy entropy and reward variance compared to self-generated trajectories (Figure 4). This demonstrates that expert anchors provide a more informative and gradient-valuable exploration space for subsequent policy optimization.

Accordingly, for each problem  $x$ , we allocate a self-explore budget  $N$  and an expert-enhanced budget  $M$ . The sampling process is formalized as three stages:

**1. Initialization.** We initialize  $\mathcal{D}_{\text{self}}$  with  $N$  starting states. For the expert trajectory  $\mathcal{H}^*$ , we measure exploration value using the step-wise policy entropy  $h_t$ , conditioned on the prefix  $\mathcal{H}_{<t}$ :

$$h_t = - \sum_{y \in Y} \pi_\theta(y|\mathcal{H}_{<t}) \log \pi_\theta(y|\mathcal{H}_{<t}) \quad (2)$$

where  $Y$  denotes the generated token sequence. We construct  $\mathcal{D}_{\text{exp}}$  by selecting the top- $k$  high-entropy prefixes, assigning each a unique tree ID  $\mathcal{T}$ .

$$\mathcal{D}_{\text{exp}} \leftarrow \text{Top-K}(\{(\mathcal{H}^{\text{new}} = \mathcal{H}_{<t}, \mathcal{T}^{\text{new}})\}_{t=2}^{|\mathcal{H}^*|}) \quad (3)$$

**2. Sampling.** To fully leverage expert experience, we sample non-terminated nodes from  $\mathcal{D}_{\text{exp}}$  at the end of each round to spawn diverse branches. The sampling probability is determined by the exploration gain  $P_t$ :

$$P_t = \frac{(h_t - h_{t-1})}{\hat{n}_{\mathcal{H}}} \cdot \mathbf{I}(|\mathcal{D}_{\text{exp}}| < M) \quad (4)$$

where  $\alpha$  is a base sampling probability and  $\hat{n}_{\mathcal{H}}$  represents the number of branches of this expert tree. If  $P_t$  exceeds a threshold  $\gamma$  and the budget  $M$  is not reached, a copy of the trajectory is created and inserted into  $\mathcal{D}_{\text{exp}}$ :

$$\mathcal{D}_{\text{exp}} \leftarrow \mathcal{D}_{\text{exp}} \cup \begin{cases} \{\text{Copy}(\mathcal{H}, \mathcal{T})\}, & \text{if } P_t > \gamma \\ \emptyset, & \text{otherwise} \end{cases} \quad (5)$$

**3. Expansion.** For all active states/trajectories  $\mathcal{H}$  in  $\mathcal{D}_{\text{self}}$  and  $\mathcal{D}_{\text{exp}}$ , we execute a single step of outward Rollout (expansion) using  $\pi_\theta$ . The experience pools are updated with the resulting longer trajectories  $\mathcal{H}_{\text{new}}$ :

$$\mathcal{D}_{\text{total}} \leftarrow \text{Update}(\{\mathcal{H}^{\text{new}} \sim \text{Rollout}(\mathcal{H}) \text{ if } |\mathcal{H}| < T\}) \quad (6)$$

We iterate Steps 2 and 3 until termination or the turn limit  $T$ . This constrained rollout ensures parallel efficiency, avoiding the unbounded expansion of MCTS (Wang et al., 2025d).

### 4.1.2 Dynamic Experience Filtering

To empower the agent to dynamically adjust its knowledge boundaries, we introduce a filtering mechanism based on gradient and performance.

First, regarding gradient validity, we always retain  $\mathcal{D}_{\text{self}}$  as the global advantage baseline. For the expert pool: if an expert tree  $\mathcal{D}_{\text{exp}}^{T_i}$  has zero reward variance  $\mathbb{V}$ , only one representative trajectory  $\mathcal{H}_i$  is kept to reduce redundancy; otherwise, the full tree is retained:

$$\mathcal{D}'_{\text{exp}} = \begin{cases} \bigcup_i \{\mathcal{H} \sim \mathcal{D}_{\text{exp}}^{T_i}\}, & \text{if } \mathbb{V}[r(\mathcal{D}_{\text{exp}}^{T_i})] = 0 \\ \bigcup_i \mathcal{D}_{\text{exp}}^{T_i}, & \text{otherwise} \end{cases} \quad (7)$$

Second, to prevent the model from regressing to suboptimal patterns, we construct the final training set  $\mathcal{D}_{\text{train}}$  based on a performance criterion. Specifically, we compare the maximum rewards yielded by self-exploration against those from expert guidance:

$$\mathcal{D}_{\text{train}} = \begin{cases} \mathcal{D}_{\text{self}}, & \text{if } r_\phi^{\text{max}}(\mathcal{D}_{\text{self}}) \geq r_\phi^{\text{max}}(\mathcal{D}'_{\text{exp}}) \\ \mathcal{D}_{\text{self}} \cup \mathcal{D}'_{\text{exp}}, & \text{otherwise} \end{cases} \quad (8)$$

This logic enforces an adaptive learning curriculum: expert trajectories are discarded when the agent matches or exceeds their performance, prioritizing the agent’s superior policy.

## 4.2 Mix Advantage and Policy Optimize

### 4.2.1 Hybrid Advantage Estimation

Upon collecting the final experience pool  $\mathcal{D}_{\text{train}}$ , the primary step is to determine the relative advantage for each trajectory  $\mathcal{H}$  in  $\mathcal{D}_{\text{train}}$ . This differentiation of relative advantage occurs at two critical levels:

**1) Global Advantage** ( $\hat{A}_{\text{global}}$ ). This metric measures the trajectory’s performance relative to the average performance across the entire pool  $\mathcal{D}_{\text{train}}$ .

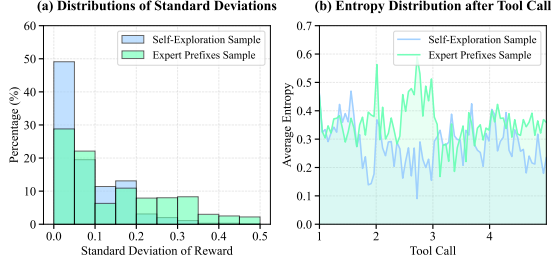


Figure 4: Distribution of Standard Deviation and Entropy under Different sampling strategies.

We leverage the global advantage for baseline estimation, which is crucial for stabilizing training:

$$\hat{A}_{\text{global}}(\mathcal{H}^i) = \frac{r_\phi(\mathcal{H}^i) - \text{mean}(r_\phi(\mathcal{H}^j))_{j=1}^{D_{\text{train}}}}{\text{mean}(r_\phi(\mathcal{H}^j))_{j=1}^{D_{\text{train}}}} \quad (9)$$

**2) Expert-Tree Relative Advantage** ( $\hat{A}_{\text{exp-tree}}$ ). We compute an internal advantage within  $\mathcal{D}_{\text{exp}}$  to measure variance among branches sharing the prefix  $\mathcal{T}$ . This guides the model to capture the expert’s granular thought patterns.

$$\hat{A}_{\text{tree}}(\mathcal{H}^i) = \frac{r_\phi(\mathcal{H}^i) - \text{mean}(r_\phi(\mathcal{H}^j))_{j=1}^{D_{\text{exp}}^{(\mathcal{T}=\mathcal{T}^i)}}}{|D_{\text{exp}}^{(\mathcal{T}=\mathcal{T}^i)}| \cdot \text{mean}(r_\phi(\mathcal{H}^j))_{j=1}^{D_{\text{exp}}^{(\mathcal{T}=\mathcal{T}^i)}}} \quad (10)$$

Since different expert trees may share the same prefix, we normalize the advantage by dividing it by the number of trajectories within the current expert tree to prevent over-optimization.

Consequently, the advantage estimation for trajectories within  $\mathcal{D}_{\text{self}}$  is solely derived from the  $\hat{A}_{\text{global}}(\cdot)$ . Conversely, for trajectories in  $\mathcal{D}_{\text{exp}}$ , the advantage estimation combines both the  $\hat{A}_{\text{global}}(\cdot)$  and  $\hat{A}_{\text{tree}}(\cdot)$ .

$$\begin{cases} \hat{A}_{\text{self}}(\mathcal{H}_i) = \hat{A}_{\text{global}}(\mathcal{H}_i) \\ \hat{A}_{\text{exp}}(\mathcal{H}_i) = \hat{A}_{\text{global}}(\mathcal{H}_i) + \hat{A}_{\text{tree}}(\mathcal{H}_i) \end{cases} \quad (11)$$

#### 4.2.2 Mix Policy Optimization

Since distribution shifts in expert trajectories cause training instability, we adopt off-policy policy shape (Yan et al., 2025) to effectively utilize the mixed experience  $\mathcal{D}_{\text{train}}$ . The policy object is:

$$\begin{aligned} \mathcal{J}_{\text{Hybrid}}(\theta) = & \frac{1}{Z} \left( \underbrace{\sum_{\mathcal{H}_i \in \mathcal{D}_{\text{self}}} \sum_{t=1}^{|\mathcal{H}_i|} \text{CLIP}(r_{i,t}(\theta), \hat{A}_i^{\text{self}}, \epsilon)}_{\text{Pure Self-Exploration (On-Policy)}} \right. \\ & \left. + \underbrace{\sum_{\mathcal{H}_k \in \mathcal{D}_{\text{exp}}} (\mathcal{L}_{\text{prefix}}(\mathcal{H}_k) + \mathcal{L}_{\text{suffix}}(\mathcal{H}_k))}_{\text{Expert prefixes and Expert guidance (Mix-Policy)}} \right) \quad (12) \end{aligned}$$

The mixed objective is divided into two parts: (1) the self-exploration part and (2) the mixed policy part. The mixed policy component, in turn, includes the prefix and suffix parts. Their specific forms are as follows:

$$\mathcal{L}_{\text{prefix}}(\mathcal{H}_k) = \sum_{t=1}^{L_{\text{prefix}}} \text{CLIP}(\rho_{k,t}(\theta), \hat{A}_k^{\text{exp}}) \quad (13a)$$

$$\mathcal{L}_{\text{suffix}}(\mathcal{H}_k) = \sum_{t=L_{\text{prefix}}+1}^{|\mathcal{H}_k|} \text{CLIP}(r_{k,t}(\theta), \hat{A}_k^{\text{exp}}) \quad (13b)$$

Where the importance sampling ratio  $r_{k,t}$  is defined at the token level  $t$ , and  $\rho_{k,t}$  is the policy reshaping function  $f(\pi_\theta) = \pi / (\pi + \lambda)$

Branching sampling creates optimization conflicts where failed explorations ( $A < 0$ ) incorrectly penalize valid shared prefixes. We address this via *Advantage-Aware Gradient Detaching*, which selectively blocks prefix gradients for failure paths ( $A < 0$ ) while retaining full updates for successful ones ( $A > 0$ ). The modified objective is:

$$\mathcal{L}_{\text{prefix}}(\mathcal{H}_k) = \sum_{t=1}^{L_{\text{pre}}} \text{CLIP}(\rho_{k,t}(\theta), \hat{A}_k^{\text{exp}}) \cdot \mathbb{I}(\hat{A}_k^{\text{exp}} > 0) \quad (14)$$

Through this asymmetric optimization strategy, we theoretically eliminate the optimization ambiguity of the shared prefix under different branches.

## 5 Experiments

### 5.1 Experiment Setup

**Tasks and Metrics.** To comprehensively evaluate the effectiveness of our E<sup>3</sup>-TIR algorithm in TIR, we conduct experiments on the following two types of tool-use tasks:

Table 1: Overall results on 10 challenging reasoning tasks. The top two results are highlighted in **bold** and underlined. The abbreviations of the datasets: Math (Math500), HQA (HotpotQA), 2Wiki. (2wikiMultiHopQA), MuSi. (MuSiQue), SimQA (SimpleQA) and Bamb (Bamboogle).

Method	Mathematical Reasoning					Knowledge-Intensive Reasoning					Avg. RMR.	
	AIME24	AIME25	AMC23	GSM8K	MATH	2Wiki.	Bamb.	HQA	MuSi.	SimQA		
<b>Qwen2.5-3B-Instruct</b>	4.5	8.6	38.3	65.3	81.0	19.2	27.5	11.5	6.3	5.9	26.8	–
+ Only SFT	6.0	10.8	35.6	58.3	81.0	30.3	35.2	35.8	16.3	19.5	32.9	7.4
+ SFT-then-RL	11.2	<u>16.5</u>	<u>51.8</u>	69.1	85.1	<b>54.2</b>	47.3	<u>50.5</u>	<u>25.3</u>	30.6	<u>44.2</u>	<b>6.1</b>
+ Zero-RL	<u>12.0</u>	13.3	48.7	<u>71.9</u>	<b>85.4</b>	46.7	<b>50.8</b>	48.8	23.2	<u>31.1</u>	43.2	7.9
+ E <sup>3</sup> -TIR (Ours)	<b>19.0</b>	<b>20.8</b>	<b>56.0</b>	<b>72.3</b>	<u>85.2</u>	<u>52.3</u>	<u>50.3</u>	<b>51.4</b>	<b>26.9</b>	<b>32.6</b>	<b>46.7</b>	<u>6.2</u>
<b>Qwen2.5-7B-Instruct</b>	7.5	9.2	50.0	74.1	89.4	18.2	29.6	15.3	7.8	6.4	30.7	–
+ Only SFT	14.6	14.1	56.2	71.3	89.6	33.3	37.9	37.1	14.6	23.2	39.2	8.2
+ SFT-then-RL	<b>25.9</b>	23.3	59.4	<u>77.4</u>	<u>91.2</u>	<b>58.4</b>	<u>53.6</u>	<u>52.3</u>	<u>27.0</u>	30.7	<u>49.9</u>	<u>7.7</u>
+ Zero-RL	21.3	<u>27.1</u>	<u>62.1</u>	76.8	90.6	55.7	49.6	50.1	24.7	<u>31.5</u>	49.0	7.9
+ E <sup>3</sup> -TIR (Ours)	<u>23.2</u>	<b>29.3</b>	<b>64.8</b>	<b>80.4</b>	<b>92.1</b>	<u>56.3</u>	<b>54.2</b>	<b>56.0</b>	<b>32.2</b>	<b>33.5</b>	<b>52.2</b>	<b>7.5</b>
<b>Llama3.1-8B-Instruct</b>	0	1.6	25.0	46.6	78.3	16.9	38.4	12.1	13.3	6.3	23.9	–
+ Only SFT	7.5	11.2	34.3	59.3	85.6	32.9	41.1	38.2	15.9	21.7	34.8	8.4
+ SFT-then-RL	<u>16.0</u>	<u>12.5</u>	52.0	<u>64.0</u>	86.1	53.8	<b>55.4</b>	<u>53.8</u>	<b>30.2</b>	<b>33.1</b>	<u>45.7</u>	<b>7.1</b>
+ Zero-RL	<b>17.2</b>	<u>12.5</u>	<u>53.7</u>	62.5	<u>87.9</u>	<u>54.3</u>	50.9	49.6	28.5	31.6	44.9	7.7
+ E <sup>3</sup> -TIR (Ours)	15.5	<b>20.0</b>	<b>58.4</b>	<b>65.2</b>	<b>88.2</b>	<b>57.4</b>	<u>55.3</u>	<b>55.6</b>	<u>29.7</u>	<u>32.7</u>	<b>47.8</b>	<u>7.4</u>

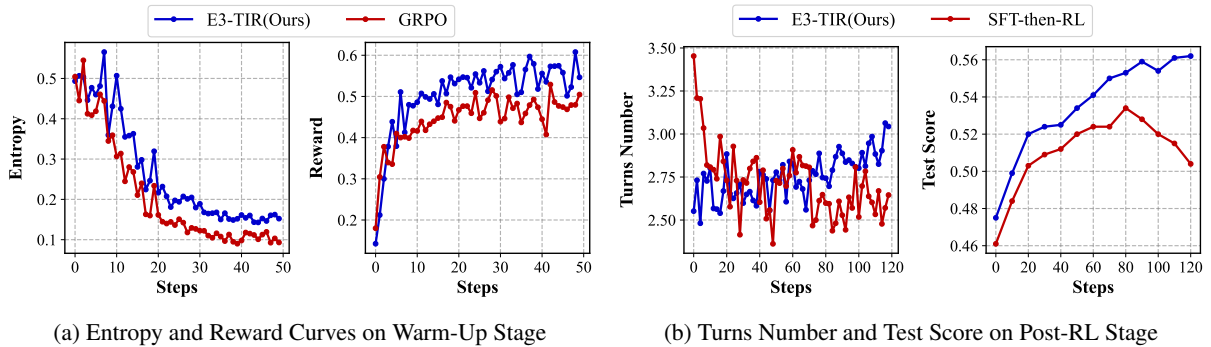


Figure 5: Comparison of the training curves on Qwen2.5-3B-Instruct.

1) **Mathematical Reasoning:**. In this task, the model can automatically generate and execute code. We use metrics such as AIME24, AIME25, MATH500 (Hendrycks et al., 2021), AMC23, and GSM8K to evaluate the model’s mathematical reasoning capabilities.

2) **Knowledge-Intensive Reasoning:** We evaluate on knowledge question benchmarks, including: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), SimpleQA (Press et al., 2023).

**Baselines.** We compare the proposed E<sup>3</sup>-TIR against the following baseline: (i) **Training Pipeline.** We compare against SFT-only, SFT-then-RL (via GRPO), Zero-RL (via GRPO), trained on the same dataset. (ii) **LLM-based Search Agent.** including Search-o1 (Li et al., 2025a), Search-R1 (Jin

et al., 2025b) and Tree-GRPO (Ding and Ye, 2025). (iii) **LLM-based Math Agent.** including ToRL (Li et al., 2025b) and SimpleTIR (Xue et al., 2025). (iv) **Mult-Tool Integrated Agent.** including Tool-Star (Dong et al., 2025b), ReCall (Chen et al., 2025b) and ARPO (Dong et al., 2025c).

**Training Guideline.** To ensure reproducibility, all training frameworks and datasets are sourced from open-access resources. Specifically, the expert trajectories data are from the Tool-Star SFT dataset. For the RL data, we also utilize the Tool-Star RL dataset.

**1. Initialization:** To mitigate reward collapse during the initial RL training stage, the model is initialized using only a small amount of self-distillation data..

**2. RL Stage:** The RL phase adopts a two-stage training paradigm:

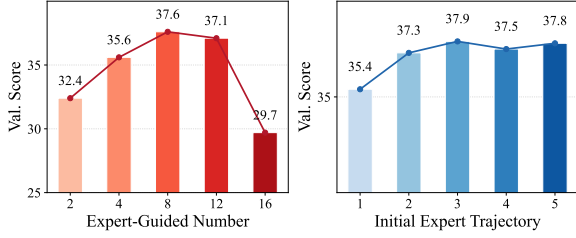


Figure 6: Ablation analysis of different hyperparameters in Expert Branch Sample.

- **E3-TIR (Warm Up).** In this stage, the model undergoes mix policy training using the SFT dataset augmented with expert experience.
- **E3-TIR (Post-RL).** In this stage, the model is trained using the standard RL dataset without any specific modifications.

The detailed introduction of training setup are listed in Appendix C.

## 5.2 Main Results

**Comparison with Training Paradigms.** Table 1 presents our main experimental results. Compared to baseline methods, E<sup>3</sup>-TIR consistently achieves superior performance. Whether applied to smaller 3B models or larger 7B/8B models, our mechanism yields a 6% average performance gain while simultaneously achieving higher tool-use efficiency.

**Comparison with SOTA Methods.** Table 2 benchmarks E<sup>3</sup>-TIR against current open-source SOTA methods. Results demonstrate that, under comparable parameter scales, our approach exhibits superior performance for both single-tool and multi-tool agents, maintaining a robust competitive edge across diverse benchmarks.

### Balancing Exploration Quality and Efficiency.

As shown in Figure 5a, E<sup>3</sup>-TIR achieves rapid reward ascent by leveraging expert anchors, in contrast to the cold start of Zero-RL. Simultaneously, the model maintains high policy entropy, indicating that it effectively exploits expert knowledge without loss diversity.

### Good Environmental Adaptability.

As shown in Figure 5b, the SFT model struggles to adapt to the multi-turn interactions, leading to declining tool usage and an overfitting “fast rise, then drop” pattern in accuracy. Conversely, E<sup>3</sup>-TIR demonstrates robust exploration with steadily increasing tool usage and maintains a stable upward performance curve throughout the process.

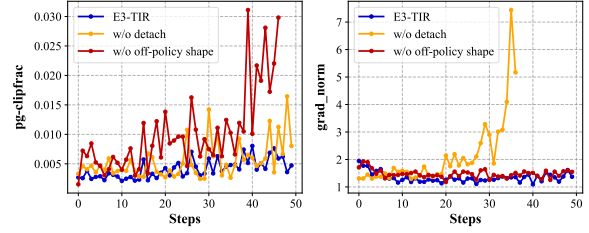


Figure 7: Ablation analysis of clip ratio and gradient norm on different mix policy optimization.

## 5.3 Ablation Study

**Component-Level Ablation.** To isolate the contribution of each core design choice, we systematically remove individual components while keeping the same expert prefixes and training budget. We additionally compare against a simple weighted SFT+RL mixture baseline (Mix SFT+RL) that replaces our loss with a weighted combination following Lv et al. (2025). Results are reported over 3 independent runs.

As shown in Table 4, removing AAGD leads to training collapse (accuracy drops to 20.6 / 14.7), confirming it is essential for resolving gradient conflicts on shared prefixes. Branching exploration contributes the largest performance gain among the remaining components: its removal causes a greater accuracy drop than removing hybrid advantage estimation, highlighting that diverse expert-anchored exploration is the primary driver of E<sup>3</sup>-TIR’s effectiveness. The Mix SFT+RL baseline also collapses (27.6 / 32.3), demonstrating that standard SFT loss is ineffective for multi-turn, long-horizon tool-use tasks due to severe distribution shifts.

**Parameter Ablation for Expert Sample.** We conducted an ablation study on two core parameters within the expert-guided sample module:

1) *Balancing Self-Exploration ( $N$ ) and Expert-Guided ( $M$ ).* We fixed the total sampling budget at 16 and varied  $M$  to evaluate model performance after the warm-up stage. As shown in Figure 6 (left), performance improves as  $M$  increases (for  $M < 8$ ), demonstrating that expert-guided trajectories effectively accelerate early convergence. However, performance deteriorates as  $M$  increases further, indicating that retaining a sufficient proportion of self-generated trajectories is critical for stabilizing training.

2) *Initial Expert Trajectories ( $K$ ).* With  $M$  fixed at 8, we varied the initial value of  $K$ . As illustrated in Figure 6 (right), performance peaks at

Table 2: Overall results on 10 challenging reasoning tasks. The top two results are highlighted in **bold** and underlined. All baseline methods use Qwen2.5-3B as the backbone. For baselines lacking a 3B Instruct version, we replicate them using the exact parameter settings from their open-source code.

Method	Computational Reasoning					Knowledge-Intensive Reasoning					Avg.
	AIME24	AIME25	AMC23	GSM8K	MATH	2Wiki	Bamb.	HQA	MuSi.	SimQA	
Qwen2.5-3B-Instruct	4.5	8.6	38.3	65.3	81.0	19.2	27.5	11.5	6.3	5.9	26.8
<b>LLM-based Math Agent</b>											
ToRL	10.0	20.0	53.0	<u>72.0</u>	84.4	24.0	25.4	17.9	8.3	6.5	32.2
SimpleTIR	17.5	19.1	<u>54.3</u>	69.4	83.9	16.1	24.2	16.5	7.2	8.8	31.7
<b>LLM-based Search Agent</b>											
Search-o1	8.6	10.3	36.5	58.6	74.2	27.4	38.5	29.5	12.5	10.8	30.7
Search-R1	3.3	10.0	31.2	58.0	60.5	44.9	44.2	50.1	22.2	<u>31.4</u>	35.6
Tree-GRPO	9.2	16.6	32.5	52.3	56.5	44.1	43.7	50.5	26.7	29.0	36.1
<b>Multi-Tool Integrated Agent</b>											
ReCall	7.6	12.4	48.2	68.6	84.1	42.6	43.7	48.7	22.6	26.5	40.5
Tool-Star	<b>20.1</b>	12.9	52.0	69.8	85.0	48.6	44.1	<b>52.4</b>	<b>27.8</b>	30.3	44.3
ARPO	16.2	<u>20.6</u>	52.0	70.2	<b>85.7</b>	<b>54.3</b>	<u>47.6</u>	50.0	26.5	29.3	<u>45.2</u>
E <sup>3</sup> -TIR (Ours)	<u>19.0</u>	<b>20.8</b>	<b>56.0</b>	<b>72.3</b>	<u>85.2</u>	<u>52.3</u>	<b>50.3</b>	<u>51.4</u>	<u>26.9</u>	<b>32.6</b>	<b>46.7</b>

Method	Knowledge.		Computational.	
	HotQA	Musique.	ACM23	MATH
E <sup>3</sup> -TIR (Warm up)	44.9	19.9	43.3	65.0
w/o $\hat{A}_{global}$	35.4 <sub>(-9.5)</sub>	13.7 <sub>(-6.2)</sub>	36.8 <sub>(-6.5)</sub>	56.3 <sub>(-8.7)</sub>
w/o $\hat{A}_{tree}$	42.6 <sub>(-2.3)</sub>	17.8 <sub>(-2.1)</sub>	43.5 <sub>(+0.2)</sub>	63.1 <sub>(-1.9)</sub>

Table 3: Ablation on Hybrid Advantages Estimation.

Method	Knowledge.		Computational.	
	HotQA	Musique.	AMC23	MATH
E <sup>3</sup> -TIR	<b>51.4</b>	<b>26.9</b>	<b>56.0</b>	<b>85.2</b>
w/o Branch	47.2 <sub>(-4.2)</sub>	23.1 <sub>(-3.8)</sub>	52.3 <sub>(-3.7)</sub>	83.6 <sub>(-1.6)</sub>
w/o AAGD	15.3 <sub>(-36.1)</sub>	8.2 <sub>(-18.7)</sub>	24.8 <sub>(-31.2)</sub>	48.6 <sub>(-36.6)</sub>
w/o Hybrid Adv.	49.5 <sub>(-1.9)</sub>	25.6 <sub>(-1.3)</sub>	55.2 <sub>(-0.8)</sub>	84.1 <sub>(-1.1)</sub>
Mix SFT+RL	22.4 <sub>(-29.0)</sub>	11.3 <sub>(-15.6)</sub>	32.5 <sub>(-23.5)</sub>	56.8 <sub>(-28.4)</sub>

Table 4: Component ablation on Qwen2.5-3B-Instruct. w/o Branch: remove expert-guided branching; w/o AAGD: remove advantage-aware gradient detaching; w/o Hybrid Adv.: remove expert-tree advantage; Mix SFT+RL: weighted SFT+RL (Lv et al., 2025).

$K = 3$ . Beyond this point, increasing  $K$  yields no significant performance variation. This is primarily attributed to the fact that the average number of interaction turns in expert trajectories typically ranges between 2 and 3.

**Effectiveness of Mix Policy Optimization.** We validate the contribution of each component through ablation studies:

1) *Mix Advantage Estimation*: Table 3 shows that  $A_{tree}$  alone causes training collapse, while the combined  $A_{tree} + A_{global}$  significantly outperforms

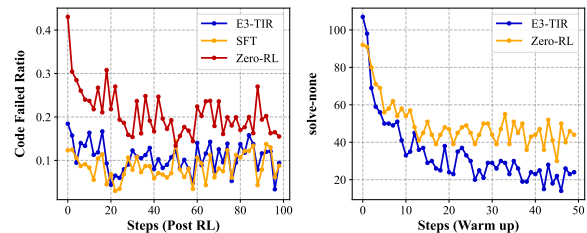


Figure 8: Comparison of Code Failed Rate and Solve-None curves across different training stages.

$A_{global}$  alone. This confirms that internal advantage captures fine-grained step differences, effectively enhancing sample efficiency.

2) *Off-policy Reshaping*: As shown in Figure 7 (left), direct updates with mixed data cause clip ratio spikes due to distributional disparity. In contrast, off-policy reshaping corrects these shifts, maintaining training stability.

3) *Shared Prefix Gradient Detach*: Figure 7 (right) demonstrates that without detachment, gradient norms oscillate due to optimization conflicts. AAGD ensures stable gradient flow, preserving the expert prefix by resolving these conflicts.

## 5.4 Analysis

**Code Execution Failure Analysis.** As shown in Figure 8 (left), we compared code failure rates during the Post-RL stage. Benefiting from the warm-up, E<sup>3</sup>-TIR exhibits a significantly lower initial error rate than Zero-RL. Notably, E<sup>3</sup>-TIR consistently maintains a low level, demonstrating superior robustness in tool usage.

Method	Avg Score	Avg Calls	Fail Rate	Redundancy
Only SFT	32.9	2.24	4.5%	0.6%
SFT-then-RL	44.2	<b>1.82</b>	<b>3.8%</b>	0.8%
Zero-RL	43.2	2.52	7.4%	0.5%
E <sup>3</sup> -TIR	<b>46.7</b>	1.97	4.0%	<b>0.6%</b>

Table 5: Tool-calling behavior audit on Qwen2.5-3B-Instruct. Avg Calls: average tool invocations per query; Fail Rate: proportion of invalid tool calls; Redundancy: ratio of repeated calls.

**Tool-Calling Behavior Audit.** Beyond task accuracy, we audit tool-calling efficiency across training paradigms. As shown in Table 5, E<sup>3</sup>-TIR achieves an average of 1.97 tool calls per query, substantially fewer than Zero-RL’s 2.52, indicating that expert anchors help establish efficient invocation patterns during the warm-up phase. The tool failure rate of E<sup>3</sup>-TIR (4.0%) is nearly half that of Zero-RL (7.4%), confirming that expert-guided exploration effectively reduces invalid syntax errors and ill-formed queries. Redundancy rates remain consistently low across all methods, suggesting that repeated tool calls are primarily driven by unresolvable code errors rather than training paradigm differences.

**Extension of Capability Boundary.** We tracked the proportion of effectively unsolvable problems during training. As illustrated in Figure 8 (right), compared to Zero-RL, E<sup>3</sup>-TIR demonstrates a continuous decline in the Solve-None rate, stabilizing at a lower final level. This confirms that expert experience effectively expands the model’s knowledge boundary, enabling it to resolve problems previously beyond its reach.

**Warm-up Stage Cost and Efficiency Analysis.** We evaluated training efficiency gains across different warm-up settings using the ROI metric (Appendix F.3). As shown in Figure 9a, ROI exhibits an “inverted-U” trend with increasing warm-up steps, attributed to the model’s diminishing reliance on experts. Meanwhile, gains from expert data size saturate after 4k samples, demonstrating that minimal expert data is sufficient for efficient startup.

**Comparison with Offline Fusion Baselines.** We further benchmark our mixed strategy against prior offline fusion methods (Yan et al., 2025; Lv et al., 2025), as illustrated in Figure 9b. While E<sup>3</sup>-TIR outperforms the off-policy free baseline by 19%, alternative methods like Luffy and HPT (utilizing

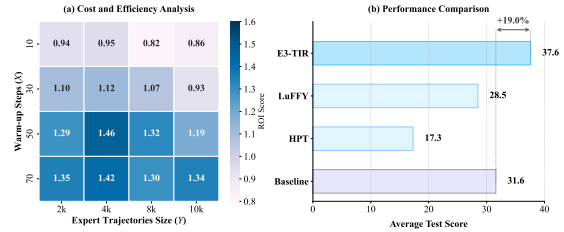


Figure 9: (a) Cost-Benefit Heatmap during the Warm-up Stage. (b) Comparison with Other Mix Strategies.

Method	ALFWorld	SCIWorld	Avg.
Only SFT	62.5	43.0	52.8
SFT-then-RL	81.5	65.0	73.3
Zero-RL	<u>86.5</u>	<u>63.5</u>	<u>75.0</u>
E <sup>3</sup> -TIR	<b>89.0</b>	<b>67.5</b>	<b>78.3</b>

Table 6: Generalization on long-horizon agent tasks using Qwen2.5-3B-Instruct. Expert data: 2K trajectories from AgentGYM (?) per task.

SFT loss) result in decreased performance. This advantage stems from the task’s complexity: robust multi-turn TIR is difficult to achieve through sequence-level optimization alone.

**Generalization on Long-Horizon Tasks.** To evaluate adaptability beyond math and QA, we conduct experiments on ALFWorld (?) and SCIWorld (?), two long-horizon agent exploration tasks requiring up to 30 interaction turns. As shown in Table 6, E<sup>3</sup>-TIR achieves the best results on both tasks (89.0 and 67.5). Notably, Zero-RL (86.5) outperforms SFT-then-RL (81.5) on ALFWorld, suggesting that SFT-induced distribution shifts can be particularly harmful in stateful environments. E<sup>3</sup>-TIR further improves upon Zero-RL by effectively leveraging expert guidance without sacrificing exploration flexibility, demonstrating its generalizability to complex, multi-step agent settings.

## 6 Conclusion

In this paper, we introduce E<sup>3</sup>-TIR, a framework that optimizes agent training by dynamically integrating different experience. By utilizing branching exploration around expert “anchors” alongside a mixed policy optimization mechanism, E<sup>3</sup>-TIR balances exploration diversity with efficiency, transforming experience utilization into proactive acquisition and establishing a new paradigm for LLMs agent training.

## Acknowledgements

This work was supported in part by National Natural Science Foundation of China (62476070), Shenzhen Science and Technology Program (JCYJ20241202123503005, GXWD20231128103232001, ZDSYS20230626091203008, KQTD20240729102154066), Department of Science and Technology of Guangdong (2024A1515011540) and National Key R&D Program of China (SQ2024YFE0200592).

## Limitations

In this section, we discuss the limitations identified in this study that point towards directions for future improvement. First, while E<sup>3</sup>-TIR significantly mitigates the dependency on massive synthetic data, it remains sensitive to the quality of initial expert prefixes. Since the core mechanism relies on branching around these “anchors”, suboptimal or low-diversity seeds may inadvertently bias the search space, thereby constraining the model’s exploration efficiency in highly complex tasks. Furthermore, our current evaluation is primarily strictly within the domains of math and QA. Consequently, the framework’s generalizability to strictly more sophisticated, real-world scenarios—particularly those necessitating the simultaneous orchestration of multiple tools or maintaining state consistency over long-horizon interactions—remains to be fully validated.

## References

- Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025a. Research: Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.
- Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025b. Research: Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.
- Yifei Chen, Guanting Dong, and Zhicheng Dou. 2025c. Toward effective tool-integrated reasoning via self-evolved preference learning. *arXiv preprint arXiv:2509.23285*.
- Zheng Ding and Weirui Ye. 2025. Treegrpo: Tree-advantage grpo for online rl post-training of diffusion models. *arXiv preprint arXiv:2512.08153*.
- 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. 2025a. Agentic entropy-balanced policy optimization. *arXiv preprint arXiv:2510.14545*.
- Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025b. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*.
- Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025c. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 6609–6625.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025a. Search-rl: Training llms to reason and leverage search engines with reinforcement learning. In *Proceedings of the Conference on Language Modeling (COLM)*.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025b. Search-rl: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-ol: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.
- Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025b. Torl: Scaling tool-integrated RL. *arXiv preprint arXiv:2503.23383*.
- Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. 2026. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *arXiv preprint arXiv:2603.16060*.

- Heng Lin and Zhongwen Xu. 2025. Understanding tool-integrated reasoning. *arXiv preprint arXiv:2508.19201*.
- Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Lixuan He, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, Ning Ding, and Bowen Zhou. 2025. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 5687–5711.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.
- Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Jirong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics (TACL)*.
- Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. 2025a. Information gain-based policy optimization: A simple and effective approach for multi-turn LLM agents. *arXiv preprint arXiv:2510.14967*.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, and Linjie Li. 2025b. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, and Linjie Li. 2025c. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.
- Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025d. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*.
- Chao Xue, Di Liang, Pengfei Wang, and Jing Zhang. 2024. Question calibration and multi-hop modeling for temporal question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19332–19340.
- Chao Xue, Yao Wang, Mengqiao Liu, Di Liang, Xingsheng Han, Peiyang Liu, Xianjie Wu, Chenyao Lu, Lei Jiang, Yu Lu, Haibo Shi, Shuang Liang, Minlong Peng, and Flora D. Salim. 2026a. Reason only when needed: Efficient generative reward modeling via model-internal uncertainty. *Preprint*, arXiv:2604.10072.
- Chao Xue, Yao Wang, Mengqiao Liu, Di Liang, Xingsheng Han, Peiyang Liu, Xianjie Wu, Chenyao Lu, Lei Jiang, Yu Lu, Haibo Shi, Shuang Liang, Minlong Peng, and Flora D. Salim. 2026b. Why supervised fine-tuning fails to learn: A systematic study of incomplete learning in large language models. *Preprint*, arXiv:2604.10079.
- Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. 2025. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*.
- Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*.
- XIANGLIN YANG, Yufei He, Shuo Ji, Bryan Hooi, and Jin Song Dong. 2026. Zombie agents: Persistent control of self-evolving LLM agents via self-reinforcing injections. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. 2025. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, and Zaibin Zhang. 2025. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.
- Pengxiang Zhao, Guangyi Liu, Yaozhen Liang, Weiqing He, Zhengxi Lu, Yuehao Huang, Yaxuan Guo, Kexin Zhang, Hao Wang, Liang Liu, and Yong Liu. 2025. Mas-bench: A unified benchmark for shortcut-augmented hybrid mobile gui agents. *Preprint*, arXiv:2509.06477.

## Appendix

<b>A</b>	<b>The Use of Large Language Models</b>	<b>12</b>
<b>B</b>	<b>Datasets and Benchmarks</b>	<b>12</b>
B.1	Mathematical Reasoning Benchmarks . . . . .	12
B.2	Knowledge-Intensive Reasoning Benchmarks . . . . .	13
B.3	Training Datasets . . . . .	13
<b>C</b>	<b>Details of Experiment Setup</b>	<b>13</b>
C.1	Supervised Fine-Tuning. . . . .	13
C.2	Reinforcement Learning. . . . .	13
C.3	Self-distillation Initialization. . . . .	13
C.4	Training Pipeline. . . . .	14
C.5	Evaluation Setup. . . . .	14
C.6	Reward Score. . . . .	14
C.7	Tool Setup. . . . .	14
<b>D</b>	<b>The Algorithm Workflow Of <math>E^3</math>-TIR</b>	<b>15</b>
<b>E</b>	<b>Theoretical Analysis</b>	<b>15</b>
E.1	Problem Formulation . . . . .	15
E.2	Failure Analysis of Zero-RL . . . . .	15
E.3	Success Guarantee of $E^3$ -TIR . . . . .	15
E.4	Numerical Illustration . . . . .	17
<b>F</b>	<b>Supplement Experimental Results</b>	<b>17</b>
F.1	Statistic Analyse of Problem in Current Training Paradigms. . . . .	17
F.2	Comparison of Expert Prefixes and Self-Exploration Sampling. . . . .	17
F.3	Warm-up Stage Cost and Efficiency Analysis . . . . .	17
F.4	Comparison with Offline Fusion Baselines . . . . .	18
F.5	Expert Trajectory Analysis during the Warm-up stage . . . . .	19
<b>G</b>	<b>Instruction Template</b>	<b>19</b>
G.1	Instructions for Mathematical Reasoning Evaluation . . . . .	19
G.2	Instructions for Knowledge-Intensive Reasoning Evaluation . . . . .	19
G.3	Instructions for Reinforcement Learning Training . . . . .	19
<b>H</b>	<b>Case Study</b>	<b>19</b>

## A The Use of Large Language Models

Throughout the preparation of this manuscript, large language models were employed exclusively for light stylistic refinement and the occasional grammatical adjustment. Every conceptual insight, analytical thread, and interpretive conclusion emerged from the authors themselves; no algorithmic assistance was solicited for the framing, design, or substance of the work, and full scientific responsibility rests with the human contributors alone.

## B Datasets and Benchmarks

### B.1 Mathematical Reasoning Benchmarks

**1. AIME24** is a dataset in evaluating the mathematical reasoning ability of models. It consists of 30 challenging math problems. All of them are from the American Invitational Mathematics Examination. The problems in the AIME24 dataset cover a wide variety of mathematical fields such as algebraic equations and geometric puzzles. Due to the difficulty characteristics and the richness of question types, it has become a popular benchmark for evaluating the reasoning performance of models, and is widely used in multiple related research experiments.

**2. AIME25** consists of 30 challenging math problems. It is directly composed of the real questions from the American Invitational Mathematics Examination newly released in February 2025. AIME25’s knowledge areas are extremely wide. It deeply covers core mathematical sections such as algebra, geometry, number theory, and combinatorial mathematics. This characteristic enables the AIME25 dataset to effectively distinguish the mathematical reasoning abilities of different models.

**3. MATH500 (Hendrycks et al., 2021)** is selected by OpenAI from the MATH evaluation dataset. It contains 500 high-difficulty math problems. These problems cover multiple mathematical fields such as algebra, geometry, calculus, and number theory. The difficulty is close to or exceeds the college level. In academic research, MATH500 dataset is often used to evaluate the performance of various reasoning models.

**4. GSM8K (?)** is an elementary school math problem dataset released by OpenAI. These problems require 2 to 8 steps to solve, mainly through a series of basic calculations to obtain the final answer. This dataset is primarily used to test the

logical and mathematical abilities of models and has been applied in multiple benchmark tests.

**5. AMC23** is a benchmark dataset designed to evaluate the advanced mathematical reasoning capabilities of large language models (LLMs). This dataset is primarily composed of problems from the 2023 American Mathematics Competitions (AMC 10/12).

## B.2 Knowledge-Intensive Reasoning Benchmarks

**HotPotQA (Yang et al., 2018)** is a question-answering dataset for multi-hop. All the documents are sourced from Wikipedia, which provides the dataset with a rich knowledge base and relatively structured information (Xue et al., 2024). It is an important benchmark for evaluating the ability of LLMs to understand complex search tasks.

**2WikiMultihopQA (Ho et al., 2020)** is a dataset specifically designed for the multi-hop question-answering task. It aims to test and evaluate the ability of natural language processing models to answer questions that require multi-step reasoning and the integration of information from different documents.

**Musique (Trivedi et al., 2022)** is a question-answering dataset specifically designed for the multi-hop question-answering task. Musique aims to be a challenging benchmark for evaluating models’ multi-hop question-answering ability. It promotes the development of models from simple information retrieval to deeper semantic understanding and logical reasoning.

**Bamboogle (Press et al., 2023)** is a diagnostic dataset designed to test the multi-hop reasoning capabilities of large language models, specifically focusing on questions that popular search engines (such as Google) cannot answer directly with a single search result. It consists of 125 manually curated questions that require “bridge” reasoning.

## B.3 Training Datasets

**Tool-Star-SFT-54K (Dong et al., 2025b)** is a supervised fine-tuning dataset designed for the "cold-start" phase of the Tool-Star framework. It contains 54,000 high-quality samples of tool-integrated reasoning, covering the complete process of autonomous tool selection, parameter invocation, and result parsing. By fine-tuning on this dataset, the model establishes foundational tool-use norms and

basic multi-step reasoning capabilities, providing a stable policy base for subsequent reinforcement learning and effectively mitigating issues like reward sparsity and excessive search space during early RL stages.

**Tool-Star-RL-10K (Dong et al., 2025b)** is a training dataset specifically tailored for the "Self-Critic Reinforcement Learning" stage, consisting of approximately 10,000 task samples. This dataset integrates various challenging tasks, supporting end-to-end interactive training in multi-tool environments. Through reinforcement exploration on these 10,000 samples, the model leverages the Self-Critic mechanism to optimize reasoning paths based on real tool feedback, significantly enhancing its logical rigor and self-correction capabilities in complex, long-horizon tasks.

## C Details of Experiment Setup

### C.1 Supervised Fine-Tuning.

In supervised fine-tuning phase, we train the model using the Llama Factory framework with a learning rate of  $7 \times 10^{-6}$ . We employ DeepSpeed ZeRO-3 and FlashAttention2 for optimization. The batch size is set to 128, with a weight decay of 0.1, and the model is trained for 3 epochs. We use BF16 mixed precision with a maximum input length of 4096 tokens.

### C.2 Reinforcement Learning.

we implement the Reinforcement Learning algorithm based on the VERL framework. As Table 7, We have provided configurations for different training methods. All of our experiments were conducted on the 8\*A100.

### C.3 Self-distillation Initialization.

To address the model’s difficulty in adhering to the specific `<python>` invocation protocol, we constructed a small-scale format alignment dataset. We sampled a subset from the SFT data, prompted the model to generate tool-integrated responses, and selected instances containing valid tool calls. We then standardized the format by replacing Markdown code blocks (`'''python ... '''`) with the target `<python> ... </python>` tags, resulting in 500 aligned examples. Since this process involved only format substitution without logical correctness verification—and the data originates from self-distillation—this stage has a negligible

Config	Warm Up	Post-RL	Zero-RL
$\gamma$	0.5	0	0
$\lambda$	0.1	0	0
n	8	16	16
m	8	0	0
k	3	0	0
optimizer	AdamW	AdamW	AdamW
learning rate	1e-6	1e-6	1e-6
training data	4000	10000	10000
total training steps	50	250	250
training batch size	128	128	128
PPO mini batch size	16	16	16
max response length	8192	8192	8192
max observation length	512	512	512
max tool-calls	4	4	4
retriever	local wiki	local wiki	local wiki

Table 7: Hyperparameters and configurations for different Training.

impact on downstream performance (Xue et al., 2026b).

#### C.4 Training Pipeline.

**E<sup>3</sup>-TIR.** In this setting, the model initially undergoes the warm-up training phase proposed in E3-TIR using the expert trajectory dataset, followed by reinforcement learning on Tool-Star10k using the Post-RL configuration.

**SFT-then-RL.** In this setting, the model first undergoes supervised fine-tuning (SFT) on the complete 52k dataset, followed by reinforcement learning on Tool-Star10k using the same Post-RL configuration.

**Zero-RL.** In this setting, the model is trained using the Zero-RL configuration. To ensure a fair comparison with E3-TIR and eliminate discrepancies arising from data coverage, we align the training data by including both Tool-Star10k and the expert dataset used in E3-TIR. However, we explicitly exclude the expert trajectory supervision (i.e., using only the prompts) from the expert dataset to strictly maintain the zero-shot exploration setting.

#### C.5 Evaluation Setup.

For evaluation, we extended the max\_obs\_length to 1k and utilized the F1 score for search tasks, while employing ground-truth matching for mathematical reasoning tasks. The specific configurations are as follows: top\_p = 0.7, temperature = 1.0, n = 4, do\_sample = True.

#### C.6 Reward Score.

Reward signals serve as the optimization objective and directly guide the behavior of the policy model during training. Formally, the overall reward  $R$  is defined as follows:

$$R = \begin{cases} \max(Acc. + r_M, Acc.) & \text{If format \& Acc.} > 0 \\ r_M & \text{If format \& Acc.} = 0 \\ -1 & \text{Otherwise} \end{cases}$$

$$r_M = \begin{cases} 0.1 & \text{If } \exists(\text{valid format}) \\ 0 & \text{Otherwise} \end{cases} \quad (15)$$

For the mathematical reasoning tasks, we score them based on the degree of match between the model’s returned answer and the actual answer. Specifically, we used the math\_verify library for the matching process. For the search task, we use the F1 score.

#### C.7 Tool Setup.

**Search Engine.** we use Wikipedia as the retrieval corpus and E5 as the retriever for four open-domain QA tasks.

**Code Interpreter.** Code snippets generated by the language model are executed in a local environment, returning either the execution result or error messages based on correctness. Our Python interpreter implementation follows the design of ToRA, ensuring secure and accurate execution.

## D The Algorithm Workflow Of $E^3$ -TIR

In this section, we provide a detailed flowchart of the  $E^3$ -TIR algorithm in diagram 1.

## E Theoretical Analysis

In this section, we provide a theoretical justification for the effectiveness of the  $E^3$ -TIR framework. Unlike token-level generation tasks, Tool-Integrated Reasoning (TIR) is modeled as a discrete decision-making process over a sequence of **nodes** (interaction turns). We analyze how Expert-Guided Branching mitigates the exploration complexity in this multi-turn setting.

### E.1 Problem Formulation

Following the definitions in Section 2.1, we consider the reasoning process as a trajectory of nodes.

**Definition E.1** (Node and Trajectory). Let a reasoning task utilize a maximum of  $T$  interaction turns. A trajectory is a sequence of nodes  $\mathcal{H} = \{u_1, u_2, \dots, u_T\}$ , where each node  $u_t = (\tau_t, a_t, o_t)$  represents the triplet of thought, action, and observation at turn  $t$ . The policy  $\pi_\theta$  generates the next node conditioned on the history of previous nodes:  $u_t \sim \pi_\theta(\cdot | \mathcal{H}_{<t})$ .

**Assumption E.2** (Step-wise Difficulty). For a complex reasoning task, obtaining a successful outcome requires a sequence of correct decisions. We assume the “student” policy  $\pi_\theta$  (before alignment) has a bounded probability  $p \ll 1$  of independently generating the optimal next node  $u_t^*$  given the correct history  $\mathcal{H}_{<t}^*$ :

$$P(u_t = u_t^* | \mathcal{H}_{<t}^*, \pi_\theta) \leq p \quad (16)$$

Here,  $p$  represents the **turn-level** success rate, not token-level probability.

### E.2 Failure Analysis of Zero-RL

In the Zero-RL setting (or pure Self-Exploration), the agent must successfully navigate from the initial state  $u_0$  to a terminal success state within  $T$  turns.

**Theorem E.3** (Curse of Horizon in Zero-RL). *For a task requiring  $T$  critical reasoning turns, the probability of a randomly initialized policy  $\pi_\theta$  generating a fully correct trajectory  $\mathcal{H}_{succ}$  is exponentially small with respect to the number of turns  $T$ .*

*Proof.* The probability of generating a complete successful trajectory is the product of the success probabilities at each node. Assuming independence for the lower bound analysis:

$$P(\text{Success} | \text{Zero-RL}) = \prod_{t=1}^T P(u_t^* | \mathcal{H}_{<t}^*, \pi_\theta) \leq p^T \quad (17)$$

Since  $p < 1$ , as the required turns  $T$  increase,  $p^T$  decays exponentially. For instance, if an agent has a 50% chance of making a correct tool call at each step ( $p = 0.5$ ) and the task requires 10 steps ( $T = 10$ ), the success rate is  $0.5^{10} \approx 0.0009$ . This explains the “Inefficient Exploration” (Figure 1) where the agent fails to collect positive feedback during the cold-start phase.  $\square$

### E.3 Success Guarantee of $E^3$ -TIR

$E^3$ -TIR addresses this challenge by utilizing **Expert-Guided Branch Sampling** (Section 4.1.1). We select an expert prefix containing  $k$  nodes, denoted as  $\mathcal{H}_{<k}^*$  (where  $k < T$ ), and perform  $G$  independent branch rollouts from this anchor node.

**Theorem E.4** (Efficiency of Node-level Branching). *Let the agent branch from an expert anchor node at turn  $k$ . The effective horizon is reduced to  $T_{eff} = T - k$  turns. With  $G$  parallel branches, the probability of discovering a successful path is significantly amplified.*

*Proof.* Given the expert history of  $k$  nodes, the agent only needs to complete the remaining  $T - k$  turns correctly.

1. **Single Branch Probability:** The probability of a single rollout successfully completing the remaining turns is:

$$P_{single} = p^{T-k} \quad (18)$$

Crucially, since we branch at deep nodes (e.g.,  $k \approx T/2$  or  $k \approx 3T/4$ ), the remaining horizon  $T - k$  is small, implying  $p^{T-k} \gg p^T$ .

2. **Group Success Probability:** With  $G$  independent branches sampled from the policy  $\pi_\theta(\cdot | \mathcal{H}_{<k}^*)$ , the probability that *at least one* branch reaches the solution is:

$$P(\text{Success} | E^3\text{-TIR}) = 1 - (1 - p^{T-k})^G \quad (19)$$

$\square$

---

**Algorithm 1** Training Workflow of  $E^3$ -TIR (Enhanced Experience Exploitation)

---

**Require:** Initial Policy  $\pi_\theta$ ; Reference Model  $\pi_{\text{ref}}$ ; Expert Dataset  $D_{\text{expert}}$ ; Self-explore Budget  $N$ ; Expert-guided Budget  $M$

**Input:** Query set  $Q$  from  $D_{\text{expert}}$

```
1: Initialize experience pools  $\mathcal{D}_{\text{self}} \leftarrow \emptyset, \mathcal{D}_{\text{exp}} \leftarrow \emptyset$ 
2: for each training iteration  $i = 1 \dots K$  do
3:   // Phase 1: Expert Experience-Based Rollout
4:   for each query  $q \in Q$  do
5:     Step 1.1: Anchor Selection
6:     Calculate step-wise entropy  $h_t$  on expert trajectory  $\mathcal{H}^*$ 
7:     Select top- $k$  high-entropy prefixes as Anchors  $\mathcal{T}_{\text{anchor}}$ 
8:     Step 1.2: Hybrid Sampling
9:     Sample  $N$  trajectories from scratch:  $\mathcal{D}_{\text{self}} \leftarrow \mathcal{D}_{\text{self}} \cup \text{Rollout}(\pi_\theta, q, N)$ 
10:    for each anchor  $t \in \mathcal{T}_{\text{anchor}}$  do
11:      Calculate exploration probability  $P_t \propto (h_t - h_{t-1})$ 
12:      if  $P_t > \gamma$  and  $|\mathcal{D}_{\text{exp}}| < M$  then
13:        Branch sample from prefix  $\mathcal{H}_{<t}^*$ :  $\mathcal{D}_{\text{exp}} \leftarrow \mathcal{D}_{\text{exp}} \cup \text{Rollout}(\pi_\theta, \mathcal{H}_{<t}^*)$ 
14:      end if
15:    end for
16:  end for
17:  // Phase 2: Dynamic Experience Filtering
18:  Discard expert trajectories with zero reward variance:  $\mathbb{V}[r] = 0$ 
19:  When the variance is 0, only one trajectory is retained.
20:  if  $\max(r(\mathcal{D}_{\text{self}})) \geq \max(r(\mathcal{D}_{\text{exp}}))$  then
21:     $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{self}}$  ▷ Self-exploration is sufficient
22:  else
23:     $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{self}} \cup \mathcal{D}_{\text{exp}}$  ▷ Need expert guidance
24:  end if
25:  // Phase 3: Mix Policy Optimization
26:  for each trajectory  $\tau \in \mathcal{D}_{\text{train}}$  do
27:    Calculate Global Advantage:  $\hat{A}_{\text{global}} \leftarrow (r(\tau) - \mu_{\text{global}}) / \sigma_{\text{global}}$ 
28:    if  $\tau \in \mathcal{D}_{\text{exp}}$  then
29:      Calculate Tree Advantage:  $\hat{A}_{\text{tree}} \leftarrow (r(\tau) - \mu_{\text{tree}}) / \sigma_{\text{tree}}$ 
30:      Final Advantage  $\hat{A} \leftarrow \hat{A}_{\text{global}} + \hat{A}_{\text{tree}}$ 
31:    else
32:      Final Advantage  $\hat{A} \leftarrow \hat{A}_{\text{global}}$ 
33:    end if
34:    Advantage-Aware Gradient Detach:
35:    if  $\tau$  shares prefix with expert AND  $\hat{A} < 0$  then
36:      Detach gradients for the shared prefix steps (Block optimization on prefix)
37:    end if
38:  end for
39:  Update  $\pi_\theta$  by maximizing  $\mathcal{J}_{\text{Hybrid}}$  with off-policy correction
40: end for
41: return Optimized Policy  $\pi_\theta$ 
```

---

**Mitigating Low-Entropy Collapse.** Traditional SFT+RL often collapses into repetitive, simple patterns (Fast Mode) to minimize turn length. By forcing the agent to branch from expert nodes

$\mathcal{H}_{<k}^*$ —which are by definition diverse and high-quality—we enforce exploration in the valid reasoning subspace. The condition for  $E^3$ -TIR to outperform Zero-RL is simply  $(1 - p^{T-k})^G > p^T$ ,

which holds true for almost any  $k > 0$  and  $G \geq 1$ .

#### E.4 Numerical Illustration

Consider a complex mathematical reasoning task requiring  $T = 8$  tool-use turns. Let the base model’s turn-level correctness probability be  $p = 0.4$ .

- **Zero-RL:** Success probability  $\approx 0.4^8 \approx 0.00065$  (approx. 1 in 1500). The reward signal is extremely sparse.
- **$E^3$ -TIR:** We use an expert prefix of  $k = 6$  turns (leaving 2 turns to solve) and sample  $G = 8$  branches.
  - Success rate per branch:  $0.4^2 = 0.16$ .
  - Total success probability:  $1 - (1 - 0.16)^8 \approx 1 - 0.25 \approx 75\%$ .

This dramatic increase from  $< 0.1\%$  to  $75\%$  demonstrates why anchoring at deep nodes allows  $E^3$ -TIR to achieve “Fast Convergence” and break the exploration bottleneck.

### F Supplement Experimental Results

#### F.1 Statistic Analyse of Problem in Current Training Paradigms.

**Zero-RL Exploration Efficiency and Mode Analysis.** We analyzed the behavioral distribution of Zero-RL during Tool-Integrated Reasoning (TIR) tasks:

- **Tool Call Failure Statistics (Fig. 2 (left,a)):** We tracked the ratio of valid tool calls and the distribution of consecutive code errors during the initial training phase (first 40 steps). A trajectory is considered valid only if it contains code blocks that are both syntactically correct and executable by the interpreter.
- **Reasoning Mode Degradation (Fig. 2 (left,b)):** We monitored the prevalence of the direct action mode (i.e., invoking tools without intermediate reasoning). We utilize the React-Mode Ratio (RMR) as a metric, defined as:

$$\text{RMR} = 10000 * \frac{N_{\text{tool}}}{L_{\text{text}}} * L_{\text{avg}} \quad (20)$$

where  $N_{\text{tool}}$  denotes the total count of tool invocations,  $L_{\text{text}}$  is the average response length and  $L_{\text{text}}$  represents the token length of the

response excluding tool generation segments. A higher RMR value implies that the agent tends to execute actions directly with minimal intermediate reasoning (Chain-of-Thought), indicating a degradation in reasoning capability.

#### SFT+RL Policy Diversity and Entropy Analysis.

To investigate the phenomenon where SFT+RL starts strong but subsequently stagnates or collapses, we conducted the following analyses:

- **Low Entropy Collapse (Fig. 2 (right,a)):** We tracked the policy entropy  $h_t$  at each RL step. Results show a sharp decline in entropy as training progresses, indicating that the model collapses into repetitive, simplistic modes and loses the capacity to explore complex reasoning paths.
- **Capability Rigidity and Distribution Shift (Fig. 2 (right,b)):** We visualized the representation of rollout sampling by embedding generated thought paths using Qwen-Embedding, followed by PCA dimensionality reduction and DBSCAN clustering. Comparing the trajectory diversity of Zero-RL and SFT models at the same training step, the results clearly show that the number of sample clusters for the SFT model decreases significantly in later stages.

#### F.2 Comparison of Expert Prefixes and Self-Exploration Sampling.

To investigate the variations in reward standard deviation and entropy under different sampling paradigms, we conducted two distinct RL training sessions using Qwen2.5-3B-Instruct, with hyperparameters fixed at  $N = 8$  and  $M = 8$ . The experiments differed in the source of the  $M$  branches: one utilized expert prefixes, while the other relied on self-generated trajectories. As this study serves as a preliminary analysis, we limited training to 50 steps and recorded:

- (1) The standard deviation of rewards within trajectory trees sharing a common prefix.
- (2) The response entropy arising from branching at different turns, calculated specifically over the first 20 generated tokens.

#### F.3 Warm-up Stage Cost and Efficiency Analysis

To quantitatively evaluate the trade-off between pre-training costs and the subsequent reinforce-

ment learning (RL) efficiency, we introduce a comprehensive metric, the **Efficient Cost-Benefit Ratio (ROI)**. This metric assesses how the E<sup>3</sup>-TIR paradigm, utilizing varying scales of expert data and warm-up durations, accelerates the convergence of the Post-RL stage while maintaining or enhancing final performance.

We formulate the training process into two distinct stages: the Warm-up stage (utilizing E<sup>3</sup>-TIR) and the Post-RL stage. The relevant variables are defined as follows:

- $X$ : The training duration (or steps) allocated to the Warm-up stage.
- $Y$ : The size of expert trajectories utilized during the Warm-up stage.
- $Z$ : The training duration (or steps) required to reach convergence in the Post-RL stage.
- $Z_{\text{base}}$ : The baseline time required for the model to reach convergence without any warm-up (i.e., Zero-RL setting).
- $V$ : The performance improvement coefficient, defined as the ratio of the final performance of the proposed method to the Zero-RL baseline (e.g.,  $V = 1.12$  implies a 12% improvement).

The ROI is designed to reward methods that achieve higher performance gains ( $V$ ) and faster total convergence ( $Z + \alpha X$ ) while minimizing reliance on extensive expert data ( $Y$ ). The formula is defined as:

$$\text{ROI} = \underbrace{V}_{\text{Performance}} \times \underbrace{\frac{Z_{\text{base}}}{Z + \alpha \cdot X}}_{\text{Time Efficiency}} \times \underbrace{1 - \frac{Y}{Y_{\text{ref}}}}_{\text{Data Efficiency}} \quad (21)$$

where:

- $\alpha$  represents the relative computational cost coefficient between the Warm-up stage and the Post-RL stage. Since the Warm-up stage (typically use Expert Prefixes) is computationally less intensive per step than RL (e.g., GRPO). Based on the update statistics of each step during the training process, we set  $\alpha = 0.62$  to normalize the warm-up cost into equivalent RL units.
- $Y_{\text{ref}}$  is the reference data scale constant (set to 52k in our experiments). This term normalizes the data efficiency, penalizing the excessive use of expert trajectories.

- Norm represents the operation of mapping this set of data to the range of 0 to 1.

**Interpretation.** A higher ROI indicates that the E3-TIR paradigm successfully leverages a small amount of expert data and a short warm-up phase to significantly reduce the expensive Post-RL training time ( $Z \ll Z_{\text{base}}$ ), thereby achieving a superior overall cost-benefit trade-off.

Data Size	Expert Trajectories			
	2k	4k	8k	10k
Warm up				
10 steps	43.0	43.4	42.7	42.9
30 steps	44.3	44.6	43.8	44.5
50 steps	43.7	46.7	46.3	46.0
70 steps	44.3	45.6	45.5	46.1

Table 8: Model performance under different expert trajectories size and Warm up steps on Qwen2.5-3B-Instruct.

Data Size	Expert Trajectories			
	2k	4k	8k	10k
Warm up				
10 steps	200	210	190	200
30 steps	180	170	160	180
50 steps	140	120	120	130
70 steps	110	100	100	110

Table 9: Post-RL convergence steps under different expert trajectories size and Warm up steps on Qwen2.5-3B-Instruct.

#### F.4 Comparison with Offline Fusion Baselines

We benchmark the mixed strategy of E3-TIR against other policy fusion paradigms by comparing it with two state-of-the-art baselines:

- **Luffy (Yan et al., 2025)**: This method formulates a dynamically adjustable objective that balances “teacher imitation” and “self-exploration.” Utilizing Group Relative Policy Optimization (GRPO), Luffy directly incorporates high-quality off-policy reasoning trajectories generated by a strong teacher (DeepSeek-R1) into the training batch, computing advantages in conjunction with the model’s own on-policy trajectories.
- **HPT (Lv et al., 2025)**: This approach implements a real-time gating mechanism based on the rollout accuracy  $P$  for each specific

query. It dynamically toggles between learning signals using a threshold  $\gamma$ : applying pure SFT with demonstrations when performance is low ( $P \leq \gamma$ ), and switching to pure GRPO for exploration when performance is high ( $P > \gamma$ ). HPT achieves a minimalist fusion—characterized as “one formula, two signals, automatic switching”—without requiring manual weight tuning or auxiliary models.

## F.5 Expert Trajectory Analysis during the Warm-up stage

We analyze the evolution of the mixed strategy during the warm-up phase, focusing on sampling ratios and relative rewards. Figure 10 (left) illustrates a decline in expert-guided samples concurrent with a rise in fully on-policy generation as training progresses. This trend reflects a reduced reliance on expert trajectories, validating E<sup>3</sup>-TIR’s two-stage training design. Additionally, Figure 10 (right) demonstrates that the mixed strategy consistently yields higher rewards than the on-policy approach, indicating that expert prefixes effectively enhance the probability of generating optimal trajectories.

## G Instruction Template

### G.1 Instructions for Mathematical Reasoning Evaluation

#### Instruction for Sampling and Reasoning Computational Datasets

You are a helpful assistant that can solve the given question step by step with the help of the python interpreter tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the python interpreter tool to calculate the math problem for fact information about specific topics if needed.

The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively. For example, `<think>` This is the reasoning process. `</think>` `<python>` python code here `</python>` `<result>` python interpreter result here `</result>` `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is `\boxed{answer here}` `</answer>`. In the last part of the answer, the final exact answer is enclosed within boxed with latex format. `{question}`

### G.2 Instructions for Knowledge-Intensive Reasoning Evaluation

#### Instruction for Sampling and Reasoning Computational Datasets

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search for fact information about specific topics if needed.

The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively, and the search query and result are enclosed within `<search>` `</search>` and `<result>` `</result>` tags respectively. For example, `<think>` This is the reasoning process. `</think>` `<search>` search query here `</search>` `<result>` search result here `</result>` `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is `\boxed{answer here}` `</answer>`.

In the last part of the answer, the final exact answer is enclosed within boxed with latex format. `{question}`

### G.3 Instructions for Reinforcement Learning Training

#### Instruction for Sampling and Reasoning Computational Datasets

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool and python interpreter tool.

Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search and python interpreter tool to calculate the math problem for fact information about specific topics if needed. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively, and the search query and result are enclosed within `<search>` `</search>` and `<result>` `</result>` tags respectively.

For example, `<think>` This is the reasoning process. `</think>` `<search>` search query here `</search>` `<result>` search result here `</result>` `<think>` This is the reasoning process. `</think>` `<python>` python code here `</python>` `<result>` python interpreter result here `</result>` `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is `\boxed{answer here}` `</answer>`. In the last part of the answer, the final exact answer is enclosed within boxed with latex format.

## H Case Study

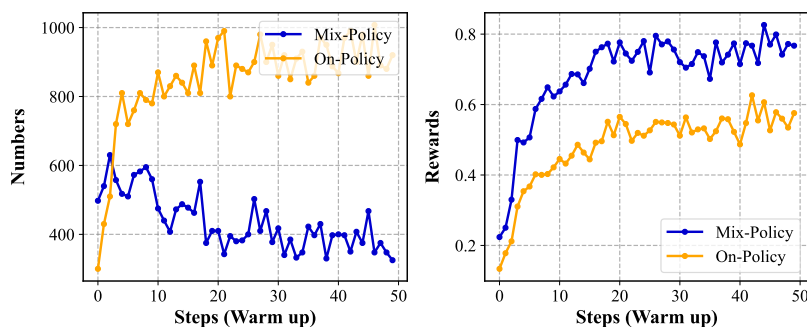


Figure 10: Variation of the sample size and rewards in the preheating stage of the mixed strategy.

Table 10: An example from E<sup>3</sup>-TIR on the Musique dataset, with special symbols used in think content, search queries, returned results and final answer highlighted with purple box, green box and red box, respectively.

Example from Musique
<p><b>Question:</b> What country has the border troops of the country of the literature of the country of Karl Maron citizenship?</p> <p><b>Labeled Answer:</b> East Germany</p>
Model Output by E <sup>3</sup> -TIR (Qwen2.5-3B)
<p>&lt;think&gt; To solve this question, I need to find out the country whose border troops have the citizenship of the country where the literature of the country of Karl Maron is from.</p> <p>Step 1: Identify the country where Karl Maron is from.</p> <p>Step 2: Identify the country whose literature is from the country of Step 1.</p> <p>Step 3: Identify the citizenship of the country from Step 2.</p> <p>Step 4: Identify the country whose border troops have the citizenship from Step 3.</p> <p>Step 1: Identify the country where Karl Maron is from.</p> <p>I'll use a query to search for this information:&lt;/think&gt;</p> <p>&lt;search&gt; Karl Maron country of citizenship &lt;/search&gt;</p> <p>&lt;result&gt; Doc 1 [id: 11966372] Title: Karl Maron Karl Maron (27 April 1903 – 2 February 1975) was a German politician, who served as the interior minister of East Germany. He also assumed different posts in East Germany's government...</p> <p>Doc 2 [id: 9746062] Title: Karl Maron ...Maron was the step-father of author Monika Maron. Karl Maron married her mother in 1955. He died in 1975. Karl Maron Karl Maron (27 April 1903 – 2 February 1975) was a German politician, who served as the interior minister of East Germany... &lt;/result&gt;</p> <p>&lt;think&gt; Step 2: Identify the country whose literature is from the country of Step 1.</p> <p>I'll use a query to search for this information:&lt;/think&gt;</p> <p>&lt;search&gt; country where the literature of East Germany is from &lt;/search&gt;</p> <p>&lt;result&gt; Doc 1 [id: 5484532] Title: Literature of East Germany Literature of East Germany East German literature is the literature produced in East Germany from the time of the Soviet occupation in 1945 until the end of the communist government in 1990...</p> <p>Doc 2 [id: 12666639] Title: Literature of East Germany the literature produced in this period among the best of the entire GDR. The 1990s saw the reunification of East and West Germany and the abrupt demise of the dream of a German "socialist utopia."... &lt;/result&gt;</p> <p>&lt;think&gt; Step 3: Identify the citizenship of the country from Step 2.</p> <p>I'll use a query to search for this information:&lt;/think&gt;</p> <p>&lt;search&gt; literature of East Germany country &lt;/search&gt;</p> <p>&lt;result&gt; Doc 1 [id: 5484532] Title: Literature of East Germany Literature of East Germany East German literature is the literature produced in East Germany...</p> <p>Doc 3 [id: 16627150] Title: Culture of East Germany Culture of East Germany The culture of East Germany varied throughout the years due to the political and historical events that took place in the 20th century... &lt;/result&gt;</p> <p>&lt;think&gt; This is last step and I need to give the final answer, which is the country whose border troops have the citizenship of the country where the literature of the country of Karl Maron is from.&lt;/think&gt;</p> <p>&lt;answer&gt;</p> <p>The final answer is <span style="border: 1px solid black; padding: 2px;">EastGermany</span></p> <p>&lt;/answer&gt;</p>

Table 11: An example from E<sup>3</sup>-TIR on the AIME25 dataset, with special symbols used in think content.

Example from Dataset
<p><b>Question:</b> Find the number of ordered pairs <math>(x, y)</math>, where both <math>x</math> and <math>y</math> are integers between <math>-100</math> and <math>100</math> inclusive, such that <math>12x^2 - xy - 6y^2 = 0</math>.</p> <p><b>Labeled Answer:</b> 117</p>
Model Output by E <sup>3</sup> -TIR (Qwen2.5-3B)
<p><b>&lt;think&gt;</b> To solve the problem, we need to find the number of ordered pairs <math>(x, y)</math> such that both <math>x</math> and <math>y</math> are integers between <math>-100</math> and <math>100</math> inclusive, and they satisfy the equation <math>12x^2 - xy - 6y^2 = 0</math>. First, let's rearrange the equation to make it easier to solve:</p> $12x^2 - xy - 6y^2 = 0.$ <p>This is a quadratic equation in terms of <math>x</math>. We can use the quadratic formula to solve for <math>x</math>:</p> $12x^2 - xy - 6y^2 = 0.$ <p>The quadratic formula for <math>ax^2 + bx + c = 0</math> is given by:</p> $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$ <p>In our equation, <math>a = 12</math>, <math>b = -y</math>, and <math>c = -6y^2</math>. So the solutions for <math>x</math> are:</p> $x = \frac{y \pm \sqrt{y^2 + 24 \cdot 6y^2}}{2 \cdot 12} = \frac{y \pm \sqrt{y^2 + 144y^2}}{24} = \frac{y \pm \sqrt{145y^2}}{24} = \frac{y \pm y\sqrt{145}}{24}.$ <p>This gives us two solutions:</p> $x = \frac{y(1 + \sqrt{145})}{24} \quad \text{and} \quad x = \frac{y(1 - \sqrt{145})}{24}.$ <p>For <math>x</math> to be an integer, <math>\frac{y(1 + \sqrt{145})}{24}</math> and <math>\frac{y(1 - \sqrt{145})}{24}</math> must be integers. Since <math>\sqrt{145}</math> is not an integer, <math>y</math> must be a multiple of 24 for <math>\frac{y(1 + \sqrt{145})}{24}</math> and <math>\frac{y(1 - \sqrt{145})}{24}</math> to be integers. Let's denote <math>y = 24k</math>, where <math>k</math> is an integer. Then the solutions for <math>x</math> become:</p> $x = \frac{24k(1 + \sqrt{145})}{24} = 1 + \sqrt{145}k \quad \text{and} \quad x = \frac{24k(1 - \sqrt{145})}{24} = 1 - \sqrt{145}k.$ <p>Since <math>\sqrt{145}</math> is not an integer, <math>k</math> must be such that <math>1 + \sqrt{145}k</math> and <math>1 - \sqrt{145}k</math> are integers. This means <math>k</math> must be an integer that makes <math>1 + \sqrt{145}k</math> and <math>1 - \sqrt{145}k</math> integers. The only way this can happen is if <math>k = 0</math>, but <math>k = 0</math> would make <math>x = 1</math>. However, we need to check if there are any other values of <math>k</math> that might work. Let's check the possible values of <math>k</math> by iterating over the range of <math>y</math> from <math>-100</math> to <math>100</math>. We will use Python to check which pairs <math>(x, y)</math> satisfy the equation. Let's write the Python code to find the number of such pairs <math>(x, y)</math>. <b>&lt;/think&gt;</b></p> <pre><b>&lt;python&gt;</b> 1 # Define the range for x and y 2 range_min = -100 3 range_max = 100 4 5 # Initialize a counter for the number of valid pairs 6 count = 0 7 8 # Iterate over all possible values of x and y 9 for x in range(range_min, range_max + 1): 10     for y in range(range_min, range_max + 1): 11         if 12*x**2 - x*y - 6*y**2 == 0: 12             count += 1 13 14 # Output the result 15 print(count) <b>&lt;/python&gt;</b></pre> <p><b>&lt;/python&gt;</b> <b>&lt;result&gt;</b>117<b>&lt;/result&gt;</b> <b>&lt;think&gt;</b> The number of ordered pairs <math>(x, y)</math>, where both <math>x</math> and <math>y</math> are integers between <math>-100</math> and <math>100</math> inclusive, such that <math>12x^2 - xy - 6y^2 = 0</math> is <span style="border: 1px solid black; padding: 2px;">117</span>. <b>&lt;/think&gt;</b> <b>&lt;answer&gt;</b>The final answer is <span style="border: 1px solid black; padding: 2px;">117</span><b>&lt;/answer&gt;</b></p>

Table 12: An example from E<sup>3</sup>-TIR on the AIME24 dataset, with special symbols used in think content.

**Example from Dataset**

**Question:** Let  $p$  be the least prime number for which there exists a positive integer  $n$  such that  $n^4 + 1$  is divisible by  $p^2$ . Find the least positive integer  $m$  such that  $m^4 + 1$  is divisible by  $p^2$ . **Labeled Answer:** 110

**Model Output by E<sup>3</sup>-TIR (Qwen2.5-3B)**

<think> To solve this problem, we need to follow these steps:

1. Identify the smallest prime number  $p$  for which there exists a positive integer  $n$  such that  $n^4 + 1$  is divisible by  $p^2$ . 2. Once we find this prime  $p$ , we need to find the smallest positive integer  $m$  such that  $m^4 + 1$  is divisible by  $p^2$ .

Let's start by iterating over prime numbers and checking the condition for  $n^4 + 1$  being divisible by  $p^2$ .

We will use Python to find the smallest prime  $p$  and then the smallest  $m$ . First, we will write a function to check if a number is prime and then iterate over prime numbers to find the smallest  $p$ . Let's write the code to achieve this. </think> <python>

```

1 from sympy import isprime, primerange
2 # Function to find the smallest prime p
3 def find_smallest_prime_p():
4     p = 2
5     while True:
6         found = False
7         # Check for the first 1000 values of n to find if there exists an n
8         # such that (n^4 + 1) is divisible by p^2
9         for n in range(1, 1000):
10            if (n**4 + 1) % (p**2) == 0:
11                found = True
12                break
13        if found:
14            return p
15    p = next_prime(p)
16
17 # Find the smallest prime p
18 smallest_prime_p = find_smallest_prime_p()
19 print((smallest_prime_p, smallest_m))

```

</python><result>Code execution failed: Traceback (most recent call last): File "<string>", line 26, in <module> File "<string>", line 15, in find\_smallest\_prime\_p NameError: name 'next\_prime' is not defined</result> <think>

It seems that the previous approach had some issues. Let's solve this problem step-by-step more methodically.

1. \*\*Identify the smallest prime  $p$  such that there exists a positive integer  $n$  for which  $n^4 + 1$  is divisible by  $p^2$ .\*\*

2. \*\*Once we find  $p$ , determine the smallest integer  $m$  such that  $m^4 + 1$  is divisible by  $p^2$ .\*\*

We will start by iterating over prime numbers and check for each prime  $p$  whether there exists an  $n$  such that  $n^4 + 1 \equiv 0 \pmod{p^2}$ . This means  $n^4 \equiv -1 \pmod{p^2}$ . We will then check for the smallest  $m$  that satisfies this condition for  $p^2$ .

Let's write the corrected and more analytical code to solve this:

1. Generate a list of prime numbers. 2. For each prime  $p$ , check if there exists an  $n$  such that  $n^4 \equiv -1 \pmod{p^2}$ . 3. Once we find the smallest such  $p$ , find the smallest  $m$  such that  $m^4 \equiv -1 \pmod{p^2}$ . </think> <python>

```

1 from sympy import isprime, primerange, next_prime
2 # Function to find the smallest m for a given prime p
3 def find_smallest_m_for_prime_p(p):
4     m = 1
5     while True:
6         if (pow(m, 4, p**2) + 1) % (p**2) == 0:
7             return m
8         m += 1
9 # Iterate over primes
10 primes = list(primerange(2, 10000))
11 for p in primes:
12     found = False
13     # Check for an n such that n^4 + 1 is divisible by p^2
14     # The search range for n can be up to p^2
15     for n in range(1, 1000):
16         if (pow(n, 4, p**2) + 1) % (p**2) == 0:
17             found = True
18             break
19     if found:
20         smallest_prime_p = p
21         break
22 smallest_m = find_smallest_m_for_prime_p(smallest_prime_p)
23 print((smallest_prime_p, smallest_m))

```

</python><result>(17, 110)</result><think>

24596

The smallest prime number  $p$  for which there exists a positive integer  $n$  such that  $n^4 + 1$  is divisible by  $p^2$  is 17. The least positive integer  $m$  such that  $m^4 + 1$  is divisible by  $17^2 = 289$  is 110. Thus, the final answer is 110. </think>

<answer>The final answer is 110</answer>