MENTOR: Reinforcement Learning via Flexible Teacher-Optimized Rewards for Tool-Use Distillation

Anonymous ACL submission

Abstract

Distilling the tool-use capabilities of large language models (LLMs) into small language models (SLMs) is essential for their practical application. The predominant approach, supervised fine-tuning (SFT), suffers from poor out-of-domain (OOD) generalization due to its rigid alignment with static teacher trajectories. While reinforcement learning (RL) offers an alternative, the capacity limitations of SLMs pose a severe dilemma: sparse outcome rewards provide insufficient guidance, whereas strict trajectory matching imposes overly restrictive constraints. To bridge this capacity-driven gap, we propose MENTOR, which introduces a flexible yet process-aware reward structure. Instead of enforcing rigid replication, MENTOR uses the teacher's reference to guide tool-use behavior, balancing behavioral alignment with downstream performance. Extensive experiments on controlled executable-tool benchmarks demonstrate that MENTOR improves OOD tool-use performance compared to SFT and strict RL baselines. Our findings suggest that within verifiable tool-use environments, flexible tool-use alignment offers a more effective approach than strict trajectory replication for developing adaptable small models.

1 Introduction

The integration of large language models (LLMs) with external tools, such as code interpreters and retrieval APIs, has enabled them as advanced agents capable of handling complex reasoning tasks (Yao et al., 2023; Paranjape et al., 2023; Wang et al., 2024b; Singh et al., 2025). However, the high inference costs of these large-scale models limit their practical use. This challenge has motivated a line of research focused on smaller language models (SLMs) to transfer the tool-assisted problem-solving capabilities of larger models (Gao et al., 2023; Gou et al., 2024; Qiu et al., 2025).

To date, knowledge distillation has relied on supervised fine-tuning (SFT), where a student model is trained to strictly replicate teacher-generated trajectories (Liu et al., 2024; Kang et al., 2025; Lyu et al., 2025). However, this rigid alignment with static token distributions stifles the exploratory autonomy of capacity-limited SLMs. Consequently, forcing exact imitation hinders adaptive tool use and severely limits generalization to out-of-domain (OOD) scenarios and to unseen tools (Luo et al., 2025b; Sun et al., 2025; Yin et al., 2025).

To mitigate reliance on static data, reinforcement learning (RL) enables models to internalize tool-use logic through interaction with the tool environment (Trung et al., 2024; Wu et al., 2025b). Unlike SFT, RL refines policies through trial and error, fostering strategies that generalize beyond predefined examples (Yu et al., 2024; Singh et al., 2025; Wu et al., 2025a). However, designing reward structures for capacity-limited SLMs poses a dilemma: outcome-only rewards provide weak guidance, leading to inefficient exploration, while strict trajectory matching imposes an overly restrictive optimization burden given their limited capacity. As a result, SLMs struggle to learn effective policy updates and underlying tool-planning logic.

To address the inherent limitations of SLM adaptation, we introduce three research questions to guide RL-based distillation in optimizing the strategic tool-use capabilities and performance of SLMs: **RQ1:** What primary failure modes characterize strategic misalignment between SLMs and LLMs during tool-integrated reasoning? **RQ2:** How can we design a reward structure that fosters flexible tool-level alignment, helping SLMs acquire teacher-guided tool-selection behavior? **RQ3:** Can this flexible, RL-based distillation effectively guide capacity-limited SLMs to achieve robust task performance and alignment with the teacher?

Our preliminary observations indicate a significant difference in tool-use strategies between LLMs and SLMs, which primarily stems from the capacity limitations of SLMs. This makes it diffi-

cult to provide proper reinforcement signals, balancing between being too uninformative and being overly strict. To address this, we propose **MENTOR** (**M**odel **E**nhancement by **N**on-rigid **T**eacher-**O**ptimized **R**ewards), a framework that mitigates this exploration failure by providing a flexible yet process-aware reward structure that bridges the strategic gap between the teacher and the student.

In our experiments across multiple tool-use benchmarks, we demonstrate that MENTOR significantly outperforms both SFT and RL baselines. Notably, our approach improves out-of-domain tool-use performance, effectively narrowing the performance gap between 1.5B–8B models and their 235B teacher. These results confirm that replacing rigid trajectory memorization with flexible, teacher-guided tool-use exploration improves SLM generalization to unseen tool-use settings.[1] To summarize, our key contributions are:

- We empirically identify the fragile tool-use behavior of capacity-limited SLMs as a primary bottleneck, creating a severe reward dilemma between providing uninformative outcome signals and imposing overly restrictive constraints.
- We propose MENTOR, a reinforcement learning framework that mitigates this dilemma by deploying a flexible yet process-aware reward structure. Our method provides moderated guidance for acquiring teacher-guided tool-use patterns while accommodating the SLM's limited capacity.
- Our experiments show that MENTOR improves out-of-domain tool-use performance across multiple benchmarks. Replacing rigid trajectory imitation with flexible tool-use distillation reduces the performance gap between SLMs and LLMs more effectively than SFT and other RL approaches.

2 Related Work

2.1 Tool-Integrated Reasoning of LLMs

Tool-integrated reasoning (TIR) enhances LLM reasoning by integrating external tools, such as code interpreters and retrieval APIs. These approaches range from prompting-based integration (Gao et al., 2023; Paranjape et al., 2023; Huang et al., 2024) to explicit fine-tuning for tool invocation (Schick et al., 2023; Kong et al., 2023; Qian et al., 2024). Recognizing that effective tool use is an inherently strategic process, recent studies have explored learning optimal invocation patterns through reinforcement learning (Yu et al., 2024; Feng et al., 2025; Qian et al., 2026). However, while large-scale models have made significant progress, extending robust tool-use policies to SLMs requires addressing distinct capacity-driven bottlenecks (Kang et al., 2025). To address these constraints, we investigate the underlying failure modes in SLM exploration and propose MENTOR, which facilitates flexible strategic alignment.

2.2 Distillation from LLMs to SLMs

To transfer advanced capabilities to SLMs, knowledge distillation typically employs supervised fine-tuning (SFT) to clone teacher-generated trajectories (Liu et al., 2024; Kang et al., 2025; Song et al., 2025; Lyu et al., 2025). Despite its prevalence, SFT-based distillation faces limitations in out-of-domain generalization, as static datasets cannot cover unseen reasoning paths (Sun et al., 2025; Yin et al., 2025). A critical consequence is that student models tend toward **imitation** by matching the token distribution of trajectories without **internalizing** the underlying strategic logic (Kandpal et al., 2023; Dai et al., 2024; Li et al., 2025). In contrast to these imitation-based methods, MENTOR utilizes an RL-based distillation framework. The teacher's trajectory serves not as a target for rigid sequential matching but as a strategic reference to guide an exploratory process aimed at fostering a more generalizable problem-solving methodology.

2.3 Reinforcement Learning for Tool-Use

Reinforcement learning (Kaelbling et al., 1996) offers a dynamic alternative to SFT by enabling models to refine their policies through environmental interaction. Recent advances, such as group relative policy optimization (GRPO), underscore that policy optimization can effectively foster self-corrective reasoning behaviors (Shao et al., 2024), with the precise design of the reward function being a decisive factor.

This design choice becomes particularly critical when considering model scale. While highly capable LLMs can discover effective policies from sparse outcome rewards (Yu et al., 2024; Feng et al., 2025; Wu et al., 2025a), capacity-limited SLMs are notably less efficient at exploration (Wei et al., 2022; Xiong et al., 2024). Existing frameworks for larger models have introduced rewards on final out-

[1] Our code is publicly available: https://anonymous.4open.science/r/MENTOR-F6E7/

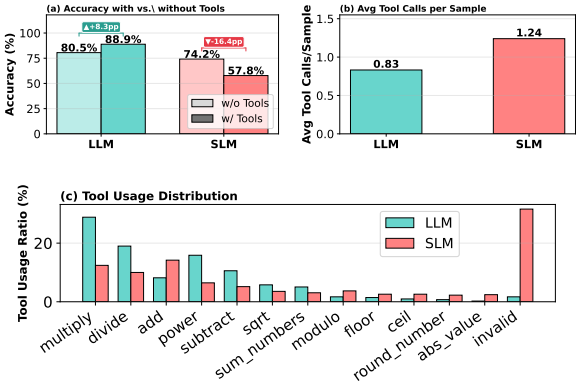

Figure 1: Comparative analysis of tool-use behavior between LLM (Qwen3-235B) and SLM (Qwen3-8B).

comes (Feng et al., 2025; Singh et al., 2025) or enforced strict alignment with ground-truth trajectories (Qian et al., 2026). However, these frameworks provide insufficient signals to induce tool use or impose an overly restrictive optimization burden. In contrast, our work leverages teacher trajectories to construct a flexible, process-aware reward signal that provides moderated yet adaptable guidance, bridging this critical optimization gap.

3 Observation: Limitations of SLM Exploration in TIR (RQ1)

To investigate the root causes of the performance gap between SLMs and LLMs, we conduct a comparative analysis of their exploration behaviors during TIR. Specifically, we compare the tool-use patterns of an LLM (Qwen3-235B) and an SLM (Qwen3-8B) (Figure 1) and analyze the learning curves of the SLM under various reward settings (Figure 2). Our analysis demonstrates that this inherent capacity gap manifests as distinct behavioral bottlenecks, posing critical optimization challenges in reinforcement learning.

Performance Disparity Despite Higher Tool-Use Frequency. Our results show that the SLM achieves a lower success rate than the teacher (Figure 1a), even though it invokes tools at a higher frequency (Figure 1b). This discrepancy suggests that the performance gap stems not from a lack of tool engagement but from the SLM's inability to employ tools effectively. Consequently, bridging this gap requires the SLM to move beyond mere tool invocation and internalize the teacher's underlying strategic logic.

Divergence in Tool Selection Patterns. This performance disparity is further evidenced by their distinct tool-selection distributions (Figure 1c). Even

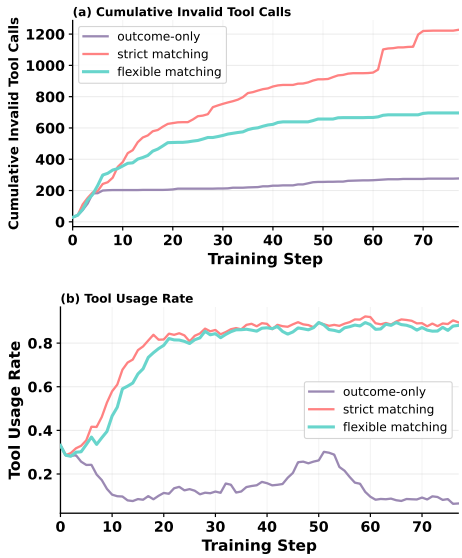

Figure 2: Comparison of various reward settings, including outcome-only, strict matching, and flexible matching (MENTOR) over training steps.

when facing identical problems, the SLM exhibits a markedly different invocation pattern compared to the LLM. While the larger model strategically focuses on complex operational tools, the student model relies heavily on simpler or redundant functions and frequently suffers from execution failures. This clear divergence indicates a fundamental gap in strategic planning capabilities, underscoring that improving TIR in SLMs requires aligning their planning logic with the advanced strategy of the teacher. Example cases contrasting LLM and SLM tool selections, along with the vulnerability from invalid tool calls, are provided in Appendix A.

The Reward Dilemma in SLM Exploration. The divergence in tool-selection patterns and high rates of execution failure create a severe initial exploration bottleneck in reinforcement learning. To investigate this, we analyze the learning curves of SLMs across various reward settings, evaluating overall tool usage rates and the cumulative number of invalid tool calls. As shown, deploying a sparse, outcome-only reward provides an insufficient reinforcement signal, causing the SLM to avoid using tools (Figure 2a). Conversely, enforcing strict trajectory matching encourages tool invocation, but the rigid sequence constraint is overly restrictive. As training continues, this strict guidance leads to a steady increase in the total number of invalid tool calls (Figure 2b). This trade-off demonstrates that overcoming exploration failure requires a balanced reward structure that both induces high tool engagement and suppresses invalid tool calls.

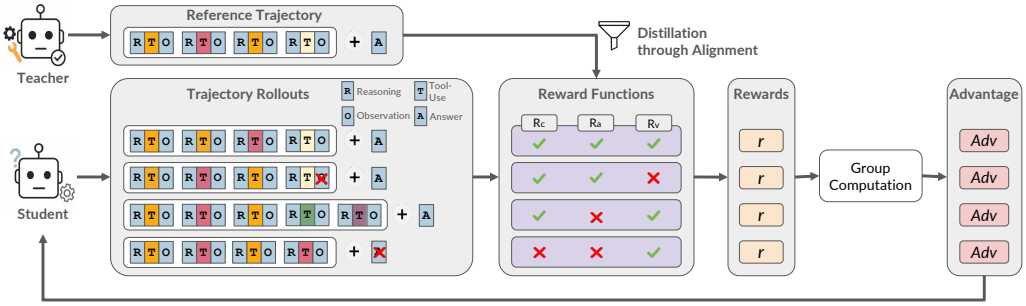

Figure 3: Overview of the MENTOR training framework. A problem-solving trajectory (τ) consists of a sequence of Reasoning (R), Tool-Use (T), and Observation (O), and a final Answer (A). Each student rollout is evaluated by a set of reward models, which generate a reward signal by aligning the student's actions against the teacher's reference trajectory.

4 Method: Designing Rewards for Tool-Use Alignment (RQ2)

To address the exploration bottlenecks and optimization challenges identified in Section 3, we introduce MENTOR, a reinforcement learning framework that provides balanced, process-aware rewards. Rather than relying solely on sparse outcome rewards or enforcing strict trajectory imitation, our method uses teacher trajectories to provide moderated guidance, enabling the student to acquire high-level tool-selection behavior guided by the teacher.

4.1 Methodology Overview

The overall process of MENTOR is illustrated in Figure 3. Given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i represents a question and y_i is its corresponding ground-truth answer, the teacher model generates an output $O^{(t)}$ for each question x. This output consists of a final answer $\hat{y}^{(t)}$ and a trajectory $\tau^{(t)}$:

$$O^{(t)} = (\tau^{(t)}, \hat{y}^{(t)}) \sim \pi_{\text{teacher}}(\cdot|x, I), \quad (1)$$

$$\tau^{(t)} = \langle (r_1, t_1, o_1), \ldots, (r_{L_\tau}, t_{L_\tau}, o_{L_\tau}) \rangle, \quad (2)$$

where I is an instruction prompt and the trajectory $\tau^{(t)}$ is a sequence of reasoning (r), tool-use (t), and observation (o) steps. Subsequently, the student model generates multiple exploratory rollouts, each denoted as $O^{(s)}$. By aligning these rollouts with the teacher's reference, MENTOR provides a balanced guidance signal that enables the student to acquire teacher-guided tool-use patterns. We employ the GRPO framework to optimize the student's policy using these signals. Detailed procedures and the objective function are provided in Appendix B.1.

4.2 Flexible Reward Design

Our reward functions are designed to mitigate the two primary trade-offs identified in Section 3: tool avoidance and continuous invalid calls. We define a composite reward signal $R(O^{(s)}, O^{(t)})$ that integrates three key components: correctness (R_c), strategic alignment (R_a), and tool validation (R_v):

$$R(O^{(s)}, O^{(t)}) = w_c R_c + w_a R_a + w_v R_v \quad (3)$$

where w_c, w_a, and w_v are hyperparameters that balance the contributions of each reward component. We provide a detailed analysis of these weights and their optimization in Appendix B.2.

Design Background. While prior frameworks have explored various reward structures, they typically frame tool-use optimization either as a sparse outcome-matching problem (Feng et al., 2025; Singh et al., 2025) or as a rigid sequence-imitation task (Qian et al., 2026). However, these empirical trade-offs suggest that neither approach is well suited to capacity-limited SLMs, which require intermediate guidance without being forced to replicate a teacher's trajectory exactly.

To bridge this gap, MENTOR shifts the optimization objective from literal sequence imitation to flexible, strategic alignment. Rather than penalizing minor structural variations or individual token mismatches, our set-based design transfers the teacher's core tool-selection strategy while allowing the student to follow alternative valid reasoning paths. This approach provides a balanced, process-aware learning environment, delivering sufficient reinforcement signals to maintain active engagement with the tool while granting the student autonomy to explore valid reasoning paths.

Final Correctness via R_c. Consistent with prior TIR frameworks (Feng et al., 2025; Singh et al., 2025; Qian et al., 2026), we include the *Correctness Reward* (R_c) to assess whether the student's final answer $\hat{y}^{(s)}$ matches the teacher's outcome $\hat{y}^{(t)}$. For robust distillation, we use only reference

trajectories in which the teacher's answer is consistent with the ground truth.

$$R_{\text{c}} = \begin{cases} 1 & \text{if } \hat{y}^{(s)} = \hat{y}^{(t)} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Internalizing Strategic Planning via R_a. To focus on strategic alignment, we introduce the *Teacher-Alignment Reward* (R_a). Rather than enforcing strict adherence to sequence formats, R_a prioritizes strategic flexibility by rewarding the student for using the same set of essential tools as the teacher. This is defined as:

$$R_a = \begin{cases} 1 & \text{if } \{t \mid t \in \tau^{(s)}\} = \{t \mid t \in \tau^{(t)}\} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

As shown in Figure 3, where the first student rollout achieves alignment despite a different tool execution order, this set-matching approach allows the student to explore alternative valid sequences without penalty. A detailed comparison with alternative approaches is provided in Appendix B.3.

Addressing Invalid Tool Calls via R_v. Since we do not enforce a strict order of tool use, the student model must ensure that all invoked tools are executed validly. To maintain the viability of execution, the *Tool Validation Reward* (R_v) penalizes non-executable or malformed tool calls in the student's trajectory $\tau^{(s)}$. This reward is granted only if every tool call in the student's trajectory executes successfully without raising an error:

$$R_{\text{v}} = \begin{cases} 1 & \forall o_i^{(s)} \in \tau^{(s)} \text{ are valid} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

5 Experiments: Effectiveness of Rewards for Tool-Use Alignment (RQ3)

5.1 Experimental Setup

Training Domain Selection. We adopt mathematical reasoning as a controlled training domain for studying reward design, as it provides well-defined success criteria and structured tool-use trajectories. Its inherent difficulty gradient drives robust exploration beyond simple policies, mitigating the risk of suboptimal convergence (Zhou et al., 2023; Wang et al., 2024a; Chen et al., 2025b; Luo et al., 2025a). This choice is also supported by prior work showing that mathematical ability is a transferable skill that provides a strong foundation for generalizable problem-solving (Wang et al., 2025; Huang et al., 2025). We use the AceReason-Math dataset for training (Chen et al., 2025b). Details of our training dataset are provided in Appendix C.1.

Models and Tools. Our teacher model is Qwen3-235B-Thinking, chosen for its strong tool-use capabilities. We use four student models to evaluate our method across different scales: Qwen3 (8B and 1.7B) and Qwen2.5 (7B and 1.5B). To augment the models with tool-use capabilities, we implement a sandbox to execute Python code on a remote server. While our primary experiments focus on the Qwen family, analyses demonstrating the extensibility of our framework to other model families are provided in Appendix D.1. Details on model versions, tools, and further implementation details are provided in Appendices C.2, C.3, and C.4.

Baselines. To evaluate our proposed framework, we compare MENTOR against four distinct baselines that represent different training approaches. **1) Vanilla SLM:** The base instruction-tuned model, serving as a performance lower bound. **2) SFT:** The predominant distillation approach, fine-tuning the student on static teacher-generated trajectories. **3) Outcome-based RL:** To isolate the impact of our flexible reward design, we employ the same RL framework but utilize a simple reward signal based solely on the final outcome. **4) Strict RL:** We also employ the same RL framework but rely on a strict sequential matching reward. Specifically, we re-implement and adapt the rigid trajectory-matching reward introduced in ToolRL (Qian et al., 2026) to fit our experimental environment. Further implementation details are provided in Appendix C.5.

Benchmarks. We evaluate our framework on a diverse set of in-domain and out-of-domain tasks to measure both task-specific performance and generalization ability. For **in-domain tasks**, our agent is evaluated on six mathematical reasoning benchmarks that provide a natural gradient of difficulty. For **out-of-domain tasks**, we assess zero-shot generalization by testing the agent on tools unseen during training. Specifically, we employ three benchmarks to evaluate retrieval-based QA and the BFCL v4 benchmark to test general tool-calling capabilities. Detailed descriptions of each dataset are provided in Appendix C.6.

Evaluation Metrics. We evaluate task performance using accuracy (Acc) and exact match (EM), where a prediction is considered correct only if it exactly matches one of the ground-truth answers after normalization. To assess how effectively the student internalizes the teacher's advanced logic, we quantify policy alignment with an **alignment**

Model	Method	Math (Acc)						
		Math-Forge	Omni-MATH	aime24	aime25	amc23	minervamath	Overall
Qwen 3	235B-Vanilla	68.00	26.00	46.67	40.00	85.00	46.69	52.06
Qwen 2.5	1.5B-Vanilla	5.18 (±0.11)	1.23 (±0.18)	0.00 (±0.00)	0.00 (±0.00)	2.50 (±2.04)	1.29 (±0.56)	1.70
	1.5B-SFT	5.79 (±0.09)	1.62 (±0.23)	1.00 (±2.25)	1.33 (±1.72)	5.75 (±2.90)	3.53 (±0.43)	3.17
	1.5B-Outcome-based	18.38 (±0.09)	3.46 (±0.18)	2.67 (±3.06)	3.00 (±2.92)	8.50* (±2.93)	4.12 (±0.34)	6.69
	1.5B-Strict RL	18.62 (±0.10)	3.78 (±0.21)	2.33 (±2.85)	3.33 (±1.95)	9.25* (±2.78)	4.38 (±0.38)	6.95
	1.5B-MENTOR	**18.84*** (±0.13)	**5.64*** (±0.28)	**10.00*** (±3.52)	**6.67*** (±2.72)	**9.75*** (±2.49)	**8.42*** (±0.44)	**9.89**
	7B-Vanilla	36.87 (±1.04)	8.60 (±1.28)	6.66 (±2.22)	5.33* (±2.81)	44.00 (±3.16)	26.48 (±0.96)	21.32
	7B-SFT	37.64 (±1.25)	11.14 (±1.99)	9.40* (±2.65)	**7.33*** (±3.44)	43.00 (±4.53)	26.38 (±2.64)	22.48
	7B-Outcome-based	50.25 (±1.32)	12.94 (±1.21)	12.00* (±3.58)	6.66* (±2.22)	45.50 (±3.07)	29.40 (±1.70)	26.13
	7B-Strict RL	50.85 (±1.41)	13.42 (±1.18)	11.67* (±3.50)	7.00* (±2.85)	46.50 (±3.20)	30.40 (±1.85)	26.64
	7B-MENTOR	**55.42*** (±1.96)	**14.72*** (±2.21)	**12.33*** (±3.52)	**7.33*** (±2.63)	**50.10*** (±2.70)	**31.86*** (±1.71)	**28.63**
Qwen 3	1.7B-Vanilla	58.70 (±0.11)	16.28 (±0.21)	25.36 (±2.32)	13.01 (±2.44)	58.00 (±4.22)	28.85 (±0.47)	33.37
	1.7B-SFT	59.99 (±0.12)	16.65 (±0.21)	31.00 (±2.25)	16.02 (±2.10)	63.75 (±3.17)	30.60 (±0.52)	36.34
	1.7B-Outcome-based	60.18 (±0.07)	17.78 (±0.28)	34.34 (±3.13)	20.00 (±2.22)	67.75 (±2.75)	31.67 (±0.48)	38.62
	1.7B-Strict RL	60.42 (±0.09)	18.34 (±0.31)	33.67 (±3.35)	20.67 (±2.34)	68.50 (±2.95)	32.05 (±0.45)	38.94
	1.7B-MENTOR	**60.66*** (±0.11)	**20.74*** (±0.22)	**36.67*** (±2.50)	**24.00*** (±4.08)	**71.00*** (±3.94)	**32.30*** (±0.38)	**40.90**
	8B-Vanilla	65.00 (±0.12)	19.46 (±0.24)	29.32 (±2.64)	20.65 (±3.44)	57.75 (±2.49)	42.55 (±0.58)	39.12
	8B-SFT	65.35 (±0.07)	20.45 (±0.21)	31.65 (±3.23)	25.65 (±4.17)	62.50 (±2.89)	43.22 (±0.31)	41.47
	8B-Outcome-based	65.48 (±0.04)	22.03 (±0.26)	35.33 (±3.22)	30.33 (±3.99)	75.50 (±2.58)	43.49 (±0.25)	45.19
	8B-Strict RL	65.76 (±0.06)	22.48 (±0.27)	35.00 (±3.40)	31.00 (±4.10)	76.50 (±2.65)	43.85 (±0.32)	45.77
	8B-MENTOR	**66.50*** (±0.11)	**24.10*** (±0.25)	**39.00*** (±3.53)	**37.00*** (±3.31)	**79.00*** (±2.69)	**43.94*** (±0.44)	**48.26**

Model	Method	BFCL-v4 (Acc)					RAG (EM)			
		Non-Live	Multi-Turn	Live	Agentic	Overall	Bamboogle	2WikiMultiHopQA	HotpotQA	Overall
Qwen 3	235B-Vanilla	87.62	51.88	82.68	18.83	50.13	41.60	42.50	34.60	39.57
Qwen 2.5	1.5B-Vanilla	68.88 (±1.15)	1.09 (±0.34)	58.99 (±0.54)	2.21 (±0.44)	24.00	0.30 (±0.31)	3.20 (±0.55)	1.47 (±0.60)	1.66
	1.5B-SFT	69.96 (±1.20)	1.58 (±0.42)	60.72 (±0.36)	2.65 (±0.29)	24.59	0.85 (±0.53)	3.91 (±0.47)	2.52 (±0.57)	2.43
	1.5B-Outcome-based	71.38 (±0.63)	1.98 (±0.50)	61.46 (±0.45)	3.09 (±0.39)	25.12	6.80 (±0.49)	10.19 (±0.41)	5.79 (±0.66)	7.59
	1.5B-Strict RL	71.75 (±0.59)	1.92 (±0.46)	61.62 (±0.42)	2.78 (±0.36)	24.85	6.92 (±0.52)	6.85 (±0.48)	5.42 (±0.62)	6.39
	1.5B-MENTOR	**72.66*** (±0.62)	**2.35*** (±0.41)	**62.03*** (±0.39)	**3.83*** (±0.40)	**25.70**	**7.36*** (±0.55)	**11.31*** (±0.54)	**6.50*** (±0.40)	**8.39**
	7B-Vanilla	72.04 (±0.33)	3.90 (±1.43)	63.78 (±0.91)	4.09 (±1.86)	26.38	14.88 (±2.48)	14.20 (±0.58)	13.08 (±1.00)	14.05
	7B-SFT	72.05 (±0.32)	4.02 (±1.08)	63.79 (±0.79)	4.27 (±1.76)	26.50	16.48 (±1.74)	13.87 (±0.53)	13.36 (±0.70)	14.57
	7B-Outcome-based	82.13 (±0.09)	14.14* (±1.69)	72.22 (±0.30)	8.15 (±1.15)	32.94	22.04 (±1.06)	17.57 (±0.55)	17.07 (±0.65)	18.89
	7B-Strict RL	82.65 (±0.14)	14.22* (±1.62)	72.62 (±0.34)	7.85 (±1.18)	33.15	23.25 (±1.10)	17.20 (±0.58)	16.85 (±0.68)	19.10
	7B-MENTOR	**83.35*** (±0.21)	**14.27*** (±1.74)	**72.81*** (±0.43)	**9.30*** (±1.34)	**33.52**	**23.56*** (±1.24)	**19.39*** (±0.40)	**19.24*** (±0.56)	**20.73**
Qwen 3	1.7B-Vanilla	80.89 (±0.70)	10.47 (±0.52)	69.90 (±0.22)	3.98 (±0.30)	29.81	14.61 (±0.53)	18.87 (±0.31)	16.11 (±0.36)	16.53
	1.7B-SFT	81.55 (±0.50)	10.45 (±0.46)	70.36 (±0.41)	4.51 (±0.39)	30.13	15.95 (±0.54)	19.69 (±0.51)	17.00 (±0.43)	17.45
	1.7B-Outcome-based	83.00 (±0.31)	11.50 (±0.38)	70.77 (±0.31)	4.73 (±0.46)	30.65	16.83 (±0.55)	20.36 (±0.53)	17.35 (±0.29)	18.18
	1.7B-Strict RL	83.55 (±0.34)	11.72 (±0.42)	70.62 (±0.32)	4.62 (±0.45)	30.95	17.65 (±0.56)	20.05 (±0.58)	17.55 (±0.32)	18.42
	1.7B-MENTOR	**83.76*** (±0.35)	**12.02*** (±0.46)	**71.05*** (±0.34)	**5.52*** (±0.60)	**31.20**	**18.28*** (±0.55)	**21.37*** (±0.60)	**18.03*** (±0.48)	**19.23**
	8B-Vanilla	87.28 (±0.49)	35.51 (±0.47)	80.16 (±0.29)	9.89 (±0.33)	41.35	32.13 (±0.64)	35.88 (±0.61)	27.47 (±0.61)	31.83
	8B-SFT	87.91 (±0.48)	36.92 (±0.60)	80.59 (±0.33)	10.58 (±0.44)	42.16	34.65 (±0.31)	37.18 (±0.36)	30.59 (±0.44)	34.14
	8B-Outcome-based	88.37 (±0.37)	38.06 (±0.29)	81.10 (±0.37)	11.03 (±0.34)	42.78	35.18 (±0.36)	37.95 (±0.30)	31.30 (±0.31)	34.81
	8B-Strict RL	88.62 (±0.38)	38.18 (±0.32)	81.32 (±0.40)	10.92 (±0.36)	42.95	35.42 (±0.38)	37.45 (±0.32)	30.92 (±0.34)	34.66
	8B-MENTOR	**89.41*** (±0.56)	**38.77*** (±0.60)	**81.96*** (±0.52)	**11.57*** (±0.42)	**43.39**	**35.77*** (±0.36)	**38.86*** (±0.56)	**31.68*** (±0.36)	**35.44**

Table 1: Main results comparing MENTOR against baselines across all evaluation benchmarks. The top table shows in-domain accuracy (%) on mathematical reasoning tasks. The bottom table shows out-of-domain performance on BFCL-v4 (accuracy %) and RAG (exact match %). Results are reported as mean (± standard deviation) over 10 runs. Overall scores are calculated differently: as a macro-average for MATH and RAG, and as an official weighted average for BFCL-v4. Values marked with an asterisk (*) denote statistical significance, based on Bonferroni-corrected pairwise test at $\alpha = 0.05$.

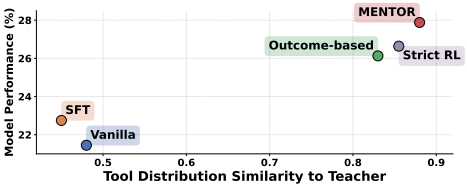

Figure 4: Correlation between task performance (Math) and alignment score (AS).

score (AS) derived from the Jensen-Shannon divergence. This score measures the divergence between a model's tool-usage distribution and the teacher's reference distribution shown in Figure 1. Further details are in Appendix C.7.

5.2 Experimental Results

The main results, presented in Table 1, demonstrate that our proposed framework, MENTOR, outperforms the baselines.

Distillation Enables Effective Tool Use. For in-domain tasks, both SFT and RL-based distillation improve tool-use effectiveness, yielding performance gains over vanilla baselines. This confirms that transferring the teacher's tool-calling capabilities serves as an effective strategy for enhancing SLM performance.

RL-Based Distillation Achieves General Performance. On out-of-domain (OOD) benchmarks, the SFT baseline shows minimal improvement, reflecting the limitations of learning from fixed trajectory distributions. While the Strict RL baseline yields notable gains over SFT, its strict sequential constraints can limit adaptability in unseen tool-use settings. By contrast, MENTOR consistently achieves the highest OOD performance across model scales. This indicates that guiding the model with structural flexibility offers a more reliable approach to generalization than enforcing rigid sequence replication through supervised learning or strict format rewards. For detailed analysis, the following subsections focus on the performance of the Qwen2.5-7B model.

5.3 Contribution of Flexible Alignment

Impact of Policy Alignment on Performance. We analyze how strategic alignment impacts downstream task performance by plotting model perfor-

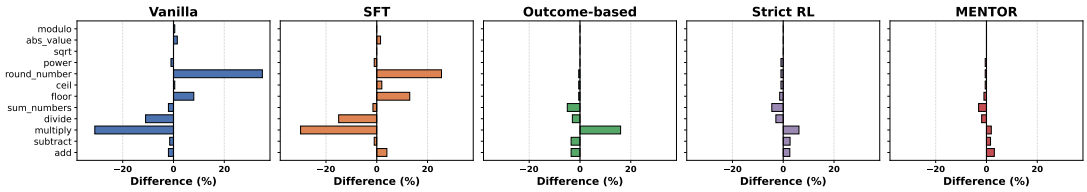

Figure 5: Comparison of tool invocation patterns between student models and the teacher. Each bar represents the percentage point difference in invocation frequency for a specific tool between the student model and the teacher.

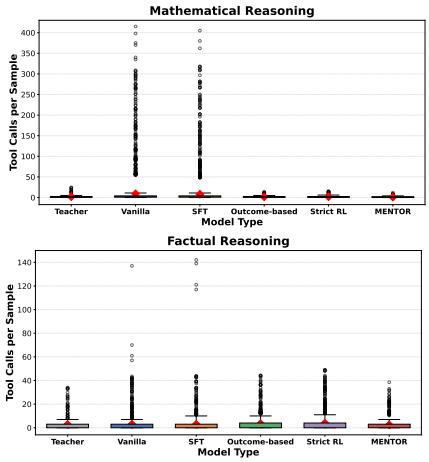

Figure 6: Tool-use efficiency on in-domain and OOD tasks, measured by the distribution of tool calls per sample.

Reward Setting	Math	BFCL	RAG	AS
(1) R_c (Outcome-based RL)	26.13	30.88	18.05	82.65
(2) $R_c + R_v$	26.80	30.46	8.35	81.84
(3) $R_a + R_v$	24.52	26.73	14.21	78.54
(4) $R_c + R_{\text{strict format}}$ (Strict RL)	26.64	29.78	15.42	86.42
(5) $R_c + R_{\text{strict format}} + R_v$	26.76	30.60	18.05	85.94
(6) $R_c + R_a + R_v$ (Ours)	**27.88**	**31.38**	**21.23**	**88.79**

Table 2: Ablation study of the reward components. Performance is measured by the overall score on the Math, BFCL, and RAG benchmarks. AS represents the alignment score.

mance against the alignment score (AS) in Figure 4. The results show a clear positive correlation between a student's behavioral similarity to the teacher and their downstream task accuracy, with the SFT baseline achieving the lowest alignment and performance. While RL-based methods generally outperform SFT, MENTOR achieves the highest accuracy and exhibits the closest teacher-aligned tool-use pattern. This simultaneous improvement in both downstream performance and distribution alignment suggests that flexible strategic guidance may be more effective for policy distillation than strict sequence replication.

Distributional Alignment of Tool Usage. To evaluate the strategic alignment of each model with the teacher, Figure 5 compares each model's tool invocation patterns with the teacher's reference policy. The SFT baseline shows significant distributional deviations, largely retaining the vanilla model's pre-existing invocation habits rather than internalizing the teacher's trajectory distribution. This divergence highlights the inherent limitations of static imitation. In contrast, reinforcement learning baselines narrow the alignment gap, though subtle variations remain observable. While Outcome-based RL exhibits skewed peaks in specific tools, the Strict RL baseline yields relatively low deviations but still leaves subtle, unresolved disparities in overall call frequencies. By contrast, MENTOR demonstrates the tightest and most balanced alignment across all tools. This suggests that MENTOR helps the student adopt tool-selection patterns closer to the teacher while still allowing trajectory-level flexibility during generation.

Behavioral Stability in Tool Usage. Figure 6 shows tool-use efficiency by plotting tool calls per question for in-domain and OOD tasks. The upper outliers and wide spreads in the Vanilla and SFT baselines highlight recurring behavioral errors, such as redundant tool-repetition loops. For in-domain tasks, RL baselines substantially reduce outliers and align with the teacher's concise distribution, indicating that RL optimization is more effective at mitigating invocation errors than SFT. However, in the OOD setting, this trend diverges, with outcome-based and strict RL baselines exhibiting wider spreads and numerous high-frequency outliers. In particular, Strict RL shows more tool repetition and higher variance than outcome-based methods. This divergence suggests that strict sequential constraints may hinder an SLM's response to failures in unseen environments. MENTOR, however, demonstrates stable variance and lower call frequency, suggesting that flexible alignment improves behavioral stability in new tasks.

5.4 Ablation Study on Reward Components

Comparative Evaluation of Reward Designs. To isolate the contribution of each reward component and compare with prior RL-based TIR frameworks, we evaluate six reward configurations. The results are shown in Table 2. Specifically, **Setting (1)** serves as the outcome-based baseline (R_c),

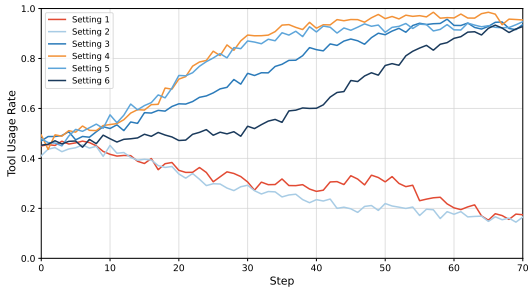

Figure 7: Tool usage rate over training steps

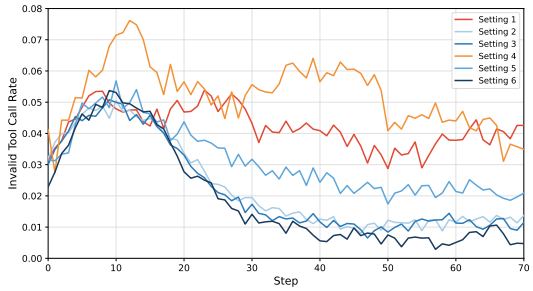

Figure 8: Invalid tool call rate over training steps

adapted from ReTool (Feng et al., 2025), while **Setting (2)** adds the tool validation reward (R_{v}) to align with the verification of ARTIST (Singh et al., 2025). **Setting (3)** evaluates strategic alignment and tool validity ($R_{\text{a}} + R_{\text{v}}$), omitting the final outcome reward (R_{c}). **Setting (4)** incorporates strict format rewards, which aligns with ToolRL (Qian et al., 2026) as our Strict RL baseline. **Setting (5)** adds R_{v} into Setting (4). Finally, **Setting (6)** is MENTOR, which employs our flexible, set-based strategic alignment (R_{a}). Note that prior reward designs are carefully reimplemented and adapted to fit our experimental framework, ensuring a fair and controlled evaluation. The results show that our comprehensive reward design (Setting 6) yields the highest overall performance across all tasks and in policy alignment.

Impact of Flexible Rewards on Alignment and Performance. The results in Table 2 show a strong positive correlation (Pearson's $r = 0.85$) between downstream performance and the alignment score (AS), demonstrating that a flexible reward design effectively optimizes both metrics simultaneously. Crucially, while the Strict RL configurations (Settings 4 and 5) achieve relatively high alignment scores, imposing strict sequential constraints yields diminishing returns on OOD tasks. In contrast, by replacing rigid rules with flexible guidance, MENTOR (Setting 6) achieves the closest policy alignment and the highest generalization accuracy across all tasks. This suggests that a flexible reward structure allows the student to balance strategic internalization with the maintenance of downstream task capabilities.

Teacher-Guided Rewards Foster Robust Tool Adoption. Figure 7 illustrates the impact of reward components on the model's tendency to utilize tools. Without the teacher's guidance (Settings 1 and 2), the sparse reward signal provides insufficient incentive to use tools, causing the student to drift toward a tool-avoidant strategy. Conversely, incorporating teacher alignment (Settings 3-6) actively encourages tool adoption. Crucially, while the strict constraints of the Rigid RL configurations (Settings 4 and 5) drive immediate but rigid usage, MENTOR (Setting 6) demonstrates a steadier learning curve that converges to high usage. This indicates that providing strategic flexibility, despite a slower initial convergence, successfully guides the SLM toward a stable and effective tool-use policy.

Validation Reward Reduces Invocation Errors. To evaluate the efficacy of our reward design in mitigating tool invocation errors, Figure 8 shows the invalid tool call rates across ablation configurations during training. The results indicate that the choice of reward components yields distinct learning behaviors. Configurations omitting the tool validation reward (R_{v}), such as Settings 1 and 4, struggle to reduce execution errors and exhibit unstable error rates throughout optimization. Conversely, configurations integrating R_{v} (Settings 2, 3, 5, and 6) rapidly suppress invalid invocations and converge to near-zero. This suggests that directly penalizing invalid calls with R_{v} is a highly effective strategy for training a more reliable agent.

6 Conclusion

To address the challenges of tool-use distillation arising from the limited capacity of SLMs, we introduce MENTOR. This framework mitigates the reward trade-off between uninformative outcome signals and overly restrictive constraints by combining reinforcement learning with a flexible reward structure. Extensive experiments across multiple reasoning benchmarks demonstrate that MENTOR improves out-of-domain tool-use performance. Our findings suggest that, when training capacity-limited SLMs, providing flexibility to learn within their operational boundaries is more effective than enforcing rigid tool replication. This approach offers a viable path to balancing strategic alignment and downstream task performance.

Limitations

While this study offers valuable insights, it is essential to acknowledge that several open challenges remain.

Extending to Other Models. Our main experiments focus on the Qwen model series (Qwen2.5 and Qwen3), whose instruction tuning and reasoning-oriented training provide a non-trivial initial ability to produce executable tool-use trajectories. This initial ability is important because our objective is to study reward design for refining and generalizing an existing tool-use policy, rather than to solve tool-call syntax grounding from scratch.

For models that lack reliable initial TIR behavior, directly applying RL can conflate two different problems: learning the basic interface for executable tool calls and optimizing the reward structure for tool-use behavior. In such cases, a short SFT warm-up is a natural initialization step that grounds the model in the tool-use format before RL-based refinement. Importantly, this warm-up is not specific to MENTOR; it should be applied uniformly to all RL-based methods when evaluating models without an initial executable tool-use policy. We therefore view the SFT-then-RL pipeline as a controlled protocol for extending reward-based tool-use distillation to weaker or non-reasoning backbones. A more systematic study of cold-start training strategies remains an important direction for future work.

Scope of Real-World Interactive Environments. Our goal in this work is to evaluate reward design for tool-use distillation under controlled executable-tool settings. Accordingly, we evaluate MENTOR in a sandbox environment where tools are invoked through executable code and where tool execution, validity, and final correctness can be measured reliably. This setup is intentionally chosen to isolate the effect of the reward structure itself, rather than to evaluate a full interactive-agent stack.

We emphasize that this scope differs from open-ended real-world interactive environments, such as web browsers, simulators, desktop interfaces, or multi-API systems. Such environments introduce additional environment-level challenges, including interface grounding, noisy observations, long-horizon state tracking, changing tool availability, recovery from failed actions, and safety constraints. These challenges can confound the evaluation of reward design, because performance differences may arise from failures in perception, state tracking, or interface control rather than from the reward signal itself.

Therefore, we leave open-ended interactive environments outside the scope of this study and focus on controlled tool-integrated reasoning settings as a clean testbed for evaluating whether capacity-aware, flexible alignment improves tool-use distillation for SLMs. Extending MENTOR to real-world interactive agents would require combining our reward design with additional mechanisms for environment grounding, action recovery, and long-horizon interaction management, which we view as a separate direction for future work.

Teacher Trajectories as Distillation References. MENTOR treats teacher trajectories as distillation references rather than globally optimal tool-use policies. We filter teacher trajectories by retaining only cases where the teacher's final answer matches the ground truth, which reduces the risk of learning from clearly incorrect demonstrations. Nevertheless, correct teacher trajectories may still be suboptimal in terms of efficiency, ordering, or minimal tool use. This issue reflects the boundary of our study: we focus on reward design for teacher-guided distillation, not on directly discovering optimal tool-use policies from ground-truth supervision or environment-level search. Within this distillation setting, teacher trajectories provide a useful behavioral prior for guiding capacity-limited SLMs, while final-correctness and tool-validation rewards prevent the student from being rewarded for invalid or unsuccessful solutions. Exploring direct optimization toward ground-truth optimal trajectories remains a complementary direction for future work.

Scope of Source-Domain Training. We adopt mathematical reasoning as the source domain because it offers one of the most suitable settings for studying tool-use distillation: it supports diverse multi-step TIR paths, provides executable tool traces, and enables reliable evaluation of both final correctness and tool validity. Domains with similarly rich and verifiable TIR trajectories could also be used to train MENTOR-style teacher-guided rewards. In our experimental setting, mathematical reasoning was the most suitable training source among the considered benchmarks because it naturally provides dense multi-step TIR trajectories, executable traces, and reliable correctness signals. We

leave systematic validation across broader source domains as future work.

Computational Constraints on Reward Weight Search. A full grid search over w_c, w_a, and w_v is computationally expensive because each configuration requires complete RL training with multiple rollouts. We therefore select the final configuration using development-set diagnostics and report sensitivity and ablation analyses in Appendix B.2. More systematic reward-weight optimization remains a direction for future work.

Ethical Statements

This work contributes to the development of efficient and capable artificial intelligence. By successfully distilling the complex tool-use capabilities of large language models (LLMs) into smaller, more efficient small language models (SLMs), MENTOR accelerates the creation of functional, on-device AI. This capability enables local deployment, reducing reliance on expensive cloud infrastructure and improving user privacy for agents that perform complex tasks, such as mathematical reasoning and information retrieval from external sources (including the web).

However, the enhanced strategic competence and tool-augmented abilities conferred by MENTOR also introduce potential risks. Since our distilled agents are capable of autonomous reasoning, web retrieval, and code execution, they could be susceptible to misuse. Potential malicious behaviors include the automated generation of harmful scripts, the execution of unauthorized actions, or the spread of misinformation via tool-based retrieval. To ensure responsible deployment, the integration of robust safeguards is essential. We emphasize that addressing these ethical and safety concerns is an important direction for future research and responsible development in the field of small language agents.

Acknowledgements

We used AI assistants, including ChatGPT[2], Gemini[3] and Grammarly[4], to support the writing and coding processes.

[2] https://chatgpt.com/
[3] https://gemini.google.com
[4] https://app.grammarly.com/

References

AI-MO. 2024. Aime. https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

Mathematical Association of America AMC. 2023. 2023 AMC 12a and 12b: American mathematics competitions. https://www.maa.org/math-competitions/amc-12. Official competition information available at the MAA website. Problem statements referenced via the Art of Problem Solving archive: https://artofproblemsolving.com/wiki/index.php/2023_AMC_12A_Problems and https://artofproblemsolving.com/wiki/index.php/2023_AMC_12B_Problems. Accessed: 2025-10-06.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, and 1 others. 2025a. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025b. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*.

Chengwei Dai, Kun Li, Wei Zhou, and Songlin Hu. 2024. Beyond imitation: Learning key reasoning steps from dual chain-of-thoughts in reasoning distillation. *arXiv preprint arXiv:2405.19737*.

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*.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *The Twelfth International Conference on Learning Representations*.

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 Neural Information Processing Systems Track*

on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual.

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 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

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

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*.

Tenghao Huang, Dongwon Jung, Vaibhav Kumar, Mohammad Kachuee, Xiang Li, Puyang Xu, and Muhao Chen. 2024. Planning and editing what you retrieve for enhanced tool learning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 975–988, Mexico City, Mexico. Association for Computational Linguistics.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 737–740. ACM.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International conference on machine learning*, pages 15696–15707. PMLR.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. 2025. Distilling llm agent into small models with retrieval and code tools. *arXiv preprint arXiv:2505.17612*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Yilun Kong, Jingqing Ruan, Yihong Chen, Bin Zhang, Tianpeng Bao, Shiwei Shi, Guoqing Du, Xiaoru Hu, Hangyu Mao, Ziyue Li, and 1 others. 2023. Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems. *arXiv preprint arXiv:2311.11315*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Preprint*, arXiv:2206.14858.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, and 1 others. 2025. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and 1 others. 2026. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, and 1 others. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2025a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *The Twelfth International Conference on Learning Representations*.

Ne Luo, Aryo Pradipta Gema, Xuanli He, Emile Van Krieken, Pietro Lesci, and Pasquale Minervini. 2025b. Self-training large language models for tool-use without demonstrations. *arXiv preprint arXiv:2502.05867*.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. 2025. From correction to mastery: Reinforced distillation of large language model agents. *arXiv preprint arXiv:2509.14257*.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

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

Cheng Qian, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2024. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 831–854, Mexico City, Mexico. Association for Computational Linguistics.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, and 1 others. 2025. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *arXiv preprint arXiv:2506.14728*.

Qwen3. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

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*.

Hoyun Song, Huije Lee, Jisu Shin, Sukmin Cho, Changgeon Ko, and Jong C. Park. 2025. Does rationale quality matter? enhancing mental disorder detection via selective reasoning distillation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21738–21756, Vienna, Austria. Association for Computational Linguistics.

Yiyou Sun, Georgia Zhou, Hao Wang, Dacheng Li, Nouha Dziri, and Dawn Song. 2025. Climbing the ladder of reasoning: What llms can-and still can't-solve after sft? *arXiv preprint arXiv:2504.11741*.

Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2024a. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. In *The Twelfth International Conference on Learning Representations*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, abs/2212.03533.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. 2025. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Haoze Wu, Yunzhi Yao, Wenhao Yu, Huajun Chen, and Ningyu Zhang. 2025a. Recode: Updating code api knowledge with reinforcement learning. *arXiv preprint arXiv:2506.20495*.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2025b. On the generalization of sft: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch every step! llm agent learning via iterative step-level process refinement. *arXiv preprint arXiv:2406.11176*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *ArXiv*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Maxwell J Yin, Dingyi Jiang, Yongbing Chen, Boyu Wang, and Charles Ling. 2025. Enhancing generalization in chain of thought reasoning for smaller models. *arXiv preprint arXiv:2501.09804*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244.

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2024. Steptool: Enhancing multi-step tool usage in llms through step-grained reinforcement learning. *arXiv preprint arXiv:2410.07745*.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and 1 others. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921*.

A Examples of Tool Failures on SLMs

A.1 Examples of Discrepancy in Tool Selection

We provide a qualitative example to illustrate the tool-use behavior discussed in Section 3. The example compares the teacher LLM and the student SLM on the same mathematical reasoning problem. It is intended to complement the aggregate results in Figure 1, rather than serve as standalone evidence.

Table 3 shows that the teacher model follows a concise tool-use trajectory. It computes $g(16) = \sqrt{16} = 4$, then $g(4) = \sqrt{4} = 2$, and finally uses the subtract tool to obtain $t(g(16)) = 3 - 2 = 1$. The tool calls directly correspond to the necessary computational steps, and the model commits to the final answer after obtaining the relevant intermediate values.

Table 4 shows a different pattern. The student model initially reaches the correct value using the same core tools, but it does not terminate the trajectory after the necessary computation is complete. Instead, it issues redundant verification calls that are themselves erroneous—swapping the arguments of one subtraction and dropping a sign in an addition—which produce conflicting values. Rather than recognizing that its original computation was already correct, the student trusts the latest, incorrect result and commits a wrong final answer. This suggests that the student struggles not only with selecting tools, but also with judging when the accumulated tool evidence is sufficient, so that unnecessary further tool use can actively corrupt an already-correct solution.

This example illustrates that the performance gap is not merely a matter of whether the student invokes tools. The student can call relevant tools and even derive the correct intermediate value, but may still struggle to decide when to stop, so that additional unnecessary tool calls introduce new errors rather than confirming the answer. These behaviors support our motivation for a reward design that jointly encourages correct task completion, teacher-guided tool selection, and valid tool execution.

A.2 The Critical Impact of Invalid Tool Calls on SLM Performance.

As illustrated in Figure 9, small language models exhibit an extreme vulnerability to tool execution failures compared to larger models. On the left, the baseline SLM generates invalid tool calls at a rate

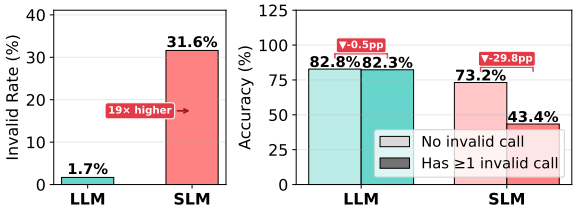

Figure 9: Invalid call rate (left) and its impact on accuracy (right).

of 31.6%, which is nearly 19 times higher than that of the teacher LLM (1.7%). More critically, as shown on the right, there is a stark contrast in error recovery capabilities between the two models. While the teacher LLM maintains its performance with a negligible accuracy drop of only 0.5 percentage points when encountering execution errors, a single invalid tool call causes the SLM's accuracy to plunge from 73.2% to 43.4%—a severe drop of 29.8 percentage points.

This catastrophic performance degradation indicates that the SLM lacks the intrinsic capability to self-correct or replan once its initial reasoning path is disrupted by a malformed tool call. Instead of diagnosing the execution error and generating a corrective alternative path, the smaller model enters a cascade of failures. Consequently, these findings underscore that bridging the performance gap in tool-integrated reasoning requires more than simply cloning optimal teacher trajectories through sequential imitation. To build true execution resilience, the SLM must internalize the underlying strategic planning logic of the teacher, enabling it to sustain coherent reasoning even when individual tool steps fail.

LLM (Qwen3-235B) Trajectory — Sample #149, MATH Algebra L3

QUESTION:

```
If t(x) = 3 - g(x) and g(x) = sqrt(x), then what is t(g(16))?

(If you have got the answer, enclose it within \boxed{} with latex format
    .)

Ground truth: 1
```

ASSISTANT (reasoning):

```
<think>
... t(g(16)) = 3 - g(g(16)). First, g(16) = sqrt(16) = 4.
Then g(4) = sqrt(4) = 2. So t(g(16)) = 3 - 2 = 1.

Looking at the tools, sqrt and subtract are available, so:
  1. Call sqrt(16)  -> 4
  2. Call sqrt(4)   -> 2
  3. Call subtract(3, 2) -> 1

So the tools needed are sqrt (twice) and subtract (once).
</think>
```

ASSISTANT (tool calls, issued in parallel in a single turn):

```
<tool_call>{"name": "sqrt",     "arguments": {"number": 16}}</tool_call>
<tool_call>{"name": "sqrt",     "arguments": {"number": 4}}</tool_call>
<tool_call>{"name": "subtract", "arguments": {"minuend": 3, "subtrahend":
    2}}</tool_call>
```

TOOL RESPONSES:

```
<tool_response>{"name": "sqrt", "arguments": {"number": 16}}
  result: 4.0
</tool_response>
<tool_response>{"name": "sqrt", "arguments": {"number": 4}}
  result: 2.0
</tool_response>
<tool_response>{"name": "subtract", "arguments": {"minuend": 3, "
    subtrahend": 2}}
  result: 1
</tool_response>
```

ASSISTANT (final answer):

```
<think>
Everything checks out. The answer should be 1.
</think>

\boxed{1}
```

Table 3: LLM trajectory for sample #149 (MATH Algebra L3). The teacher (Qwen3-235B) plans the full computation in a single reasoning step, issues the three required tool calls (`sqrt`, `sqrt`, `subtract`) in parallel within one turn, and commits the boxed answer as soon as the results return. This concise, decisive pattern contrasts with the SLM in Table 4, which reaches the same value but then derails into redundant, erroneous verification calls and commits a wrong final answer.

SLM (Qwen3-8B) Trajectory — Sample #149, MATH Algebra L3

QUESTION:

```
If t(x) = 3 - g(x) and g(x) = sqrt(x), then what is t(g(16))?
Ground truth: 1
```

Initial solve (turns 1–3): three tool calls correctly yield 1.0.

```
<think> g(16) = sqrt(16). Let me check. </think>
<tool_call>{"name": "sqrt", "arguments": {"number": 16}}</tool_call>
              -> 4.0

<think> Now g(4.0) = sqrt(4.0) = 2.0. </think>
<tool_call>{"name": "sqrt", "arguments": {"number": 4.0}}</tool_call>
             -> 2.0

<think> So t(4.0) = 3 - 2.0 = 1.0. </think>
<tool_call>{"name": "subtract", "arguments": {"minuend": 3, "subtrahend":
    2.0}}</tool_call>  -> 1.0
```

The correct answer (1.0) is now available. A confident agent would commit `\boxed{1}` *here. The SLM does not.*

Verify-loop (turns 4–5): two redundant calls, issued out of self-doubt, are themselves erroneous.

```
<think> Let me double-check the subtraction the other way. </think>
<tool_call>{"name": "subtract", "arguments": {"minuend": 2.0, "subtrahend
    ": 3}}</tool_call>  -> -1.0  (arguments swapped: WRONG)

<think> That gives -1.0, not 1.0. Let me try once more with add. </think>
<tool_call>{"name": "add", "arguments": {"firstNumber": 3, "secondNumber":
     2.0}}</tool_call>  -> 5.0  (sign error, should be -2.0: WRONG)
```

Final turn (turn 6) — commits the wrong value:

```
<think> My checks disagree (1.0, -1.0, 5.0); the latest gave 5.0. </think>

\boxed{5}
```

Final answer submitted: 5 — incorrect (ground truth is 1). The model abandoned its already-correct result (1.0) and trusted a later, erroneous tool call.

Table 4: SLM trajectory for sample #149 (condensed; `<think>` blocks are shortened to the model's key sentence per turn). The student derives the correct value (1.0) at turn 3 but does not commit. Instead, driven by self-doubt rather than computational need, it issues two redundant verification calls that are themselves erroneous — swapping the arguments of `subtract` (yielding -1.0) and dropping a sign in `add` (yielding 5.0). Faced with the conflicting results, it abandons its already-correct answer and commits the latest, wrong value (`\boxed{5}`), so the sample is marked incorrect. Total: 6 turns, 5 tool calls (2 redundant and erroneous). This verify-loop pattern contributes additional `add`/`subtract` calls to the SLM's tool distribution shown in Figure 1, and illustrates how unnecessary tool use can actively corrupt an already-correct solution.

B Details on MENTOR

B.1 Adapting GRPO for Reasoning Distillation

We adapt the GRPO algorithm to distill the teacher's problem-solving methodology by configuring the teacher's successful trajectory as a high-reward target. This reward-driven setup trains the student to internalize the teacher's strategic tool-use policy, rather than merely imitating a fixed sequence of actions. This process is detailed in Algorithm 1.

Algorithm 1 Training with GRPO

Require: Student model π_θ, old student model π_{old}, Teacher model π_{teacher}, task dataset $\mathcal{D}$, group size G, indicator function I

```
for each training iteration do
    for each question x_i do
        Generate reference trajectory O^(t) from π_teacher
        Sample G rollouts {O_1^(s), ..., O_G^(s)} from π_old
        for each rollout O_j^(s) do
            Compute outcome rewards R(O_j^(s), O^(t))
        end for
        Compute groupwise advantages Â_{j,k} for all O_j^(s)
        Apply indicator I to mask tool output tokens
        Compute GRPO loss L_GRPO and update π_student
    end for
end for
```

Generating Rollouts For each question x_i in our training set, we perform two simultaneous generation steps. First, we retrieve the corresponding reference trajectory, $(\tau^{(t)}, \hat{y}^{(t)})$, which was generated by the teacher model as described in Section 4.1. Then, we use the current student model (π_{old}) to generate a group of G rollouts. This step is crucial as it allows the student to explore diverse reasoning paths and tool-use strategies for the same problem, providing the varied data needed for the GRPO comparison.

Student Policy Optimization The core idea of our proposed method is to update a policy by aligning a high-quality reference against a group of sampled candidates. We optimize the tool-use policy of the student model based on the reference trajectory generated by the teacher. We optimize the student model by maximizing the objective function as shown in Equation 7, where ϵ and β are hyperparameters for clipping and KL regularization, respectively. The advantage, $\hat{A}_{j,k}$, is computed from the relative rewards within the sample group. The reference policy π_{ref} used for KL regularization is the initial student model before RL training. $I(\tau_{j,k})$ is an indicator function used for loss masking, which equals 1 if $\tau_{j,k}$ is an LLM-generated token, and 0 otherwise.

B.2 Reward Weight Optimization

The composite reward in Equation 3 is scaled by three hyperparameters, w_c, w_a, and w_v, which balance the relative contributions of correctness, strategic alignment, and tool validation signals, respectively. Because each reward configuration requires full RL training with multiple rollouts, exhaustive weight search is computationally expensive. We therefore adopt an empirical tuning protocol to determine the hyperparameters under these resource limitations.

Tuning Protocol. We monitor the values of each reward component throughout the training process, alongside the overall learning curve. We adjust the weights under two specific conditions: (i) when a single reward component becomes overly dominant and suppresses the signals from the other components, or (ii) when training becomes unstable, as indicated by fluctuating tool usage or a sudden increase in invalid calls. The goal of this tuning is not to maximize any single metric, but to maintain a balanced reward system where all three signals jointly guide the model's policy.

Sensitivity Analysis. To validate our final configuration of $(w_c, w_a, w_v) = (0.7, 0.3, 0.1)$, we conduct a controlled sensitivity sweep by varying each weight independently while holding the other two fixed, alongside a uniform-weight baseline for comparison. The results are presented in Table 5, where Math, BFCL, and RAG represent the overall task scores, and AS denotes the alignment score.

Observations. The sensitivity analysis reveals three key trade-offs that justify our final hyperparameter selection. First, along the correctness axis (w_c), increasing w_c to 1.0 provides no further benefits for Math (-0.19) and significantly degrades both OOD performance (-0.44 on BFCL, -1.31 on RAG) and policy alignment (-2.08 AS). This indicates that correctness optimization is already saturated at $w_c = 0.7$, and further amplification only weakens the strategic alignment signal. Conversely, reducing w_c to 0.5 causes the largest in-domain performance drop (-1.04 on Math) while leaving the alignment score relatively unchanged (-0.58 AS). This confirms that task correctness is the primary driver of task success.

Second, along the teacher alignment axis (w_a), decreasing w_a to 0.1 results in the most severe alignment degradation in the table (-2.36 AS),

$$\mathcal{J}_{GRPO}(\theta) = E_{x_i \sim \mathcal{D},\, \{O_j^{(s)}\}_{j=1}^{G} \sim \pi_{old}(\cdot|x_i)} \Bigg[\frac{1}{G} \sum_{j=1}^{G} \left(\sum_{k=1}^{|\tau_j^{(s)}|} I(\tau_{j,k}) \right)^{-1} \sum_{k=1}^{|\tau_j^{(s)}|} \min \left(\frac{\pi_\theta(\tau_{j,k}|\tau_{j,<k}, x_i)}{\pi_{old}(\tau_{j,k}|\tau_{j,<k}, x_i)} \hat{A}_{j,k}, clip(\frac{\pi_\theta(\tau_{j,k}|\tau_{j,<k}, x_i)}{\pi_{old}(\tau_{j,k}|\tau_{j,<k}, x_i)}, 1-\epsilon, 1+\epsilon) \hat{A}_{j,k} \right) \cdot I(\tau_{j,k}) - \beta D_{KL}[\pi_\theta || \pi_{ref}] \Bigg] \tag{7}$$

(w_c, w_a, w_v)	Math	BFCL	RAG	AS
Reference (ours)				
(0.7, 0.3, 0.1)	**27.88**	**31.38**	**21.23**	88.79
Varying w_c (correctness)				
(1.0, 0.3, 0.1)	27.69	30.94	19.92	86.71
(0.5, 0.3, 0.1)	26.84	30.75	20.22	88.21
Varying w_a (teacher alignment)				
(0.7, 0.1, 0.1)	27.32	30.51	20.04	86.43
(0.7, 0.5, 0.1)	27.31	31.02	20.54	**88.95**
Varying w_v (tool validation)				
(0.7, 0.3, 0.0)	26.97	30.18	18.62	87.02
(0.7, 0.3, 0.3)	27.51	31.08	20.86	88.34
Uniform reference				
(0.33, 0.33, 0.33)	27.06	30.74	20.41	88.05

Table 5: Sensitivity of MENTOR to the reward weights (w_c, w_a, w_v). Each group perturbs a single axis while holding the others fixed; the final row applies uniform weights as an additional reference. Math, BFCL, and RAG report the overall task scores; AS is the alignment score (Appendix C.7).

alongside a notable decrease on BFCL (-0.87). Pushing w_a to 0.5 improves the alignment score by $+0.16$ AS but trades away task accuracy across all benchmarks. This directly mirrors the well-known alignment-versus-utility trade-off, confirming that our moderate setting of $w_a = 0.3$ serves as a balanced joint optimum.

Third, along the tool validation axis (w_v), setting w_v to 0.0 produces the lowest RAG score in the table (-2.61) and a clear drop on BFCL (-1.20). This drop is consistent with the higher rate of execution failures observed when validation rewards are absent. However, increasing w_v to 0.3 yields no additional task performance gains, indicating that a small but non-zero penalty is sufficient to suppress malformed tool calls without making the model's policy overly conservative.

Finally, the uniform-weight baseline $(0.33, 0.33, 0.33)$ achieves a comparable alignment score (-0.74 AS) but underperforms the reference configuration on every task metric. This reinforces that our final configuration is not dominated by a single component, but rather emerges from the joint balance of all three signals.

We also note that the main ablation study in Table 2 implicitly evaluates extreme weight configurations: Settings (1)–(2) correspond to $w_a = 0$ and Setting (3) corresponds to $w_c = 0$. Because both extremes underperform MENTOR (Setting 6), these results reinforce the conclusion that all three reward components play complementary, non-redundant roles in policy distillation.

B.3 Comparison of the Set-based Approach with Alternative Methods

Motivation for Flexible Teacher Alignment. The teacher-alignment reward is designed to provide intermediate guidance between the two undesirable extremes observed in Section 3: sparse outcome-only rewards, which provide insufficient incentive for tool use, and strict trajectory matching, which can over-constrain capacity-limited SLMs. Our goal is therefore not to force the student to reproduce the teacher's trajectory exactly, but to provide a flexible alignment signal that encourages the student to use the teacher-selected tools while allowing the execution order and trajectory structure to adapt to the student's capacity. This motivates a reward that is permissive to order-level variations in how the same teacher-selected tools are executed, while remaining informative enough to distinguish trajectories that preserve the teacher's high-level tool-selection behavior from those that drift toward tool avoidance or unstable tool invocation.

Alternative Flexible Alignment Rewards. The set-based reward R_a used in MENTOR is one instantiation of a broader family of flexible teacher-alignment rewards. To examine whether the benefit comes from the general principle of flexible alignment or from the specific set-matching formulation, we compare five variants that differ in order sensitivity, multiplicity, and reward granularity. Let $\tau^{(s)}$ and $\tau^{(t)}$ denote the student and teacher tool sequences, $S(\cdot)$ extract the set of distinct tools, and $M(\cdot)$ extract the multiset of tools with call counts.

- **Set matching (MENTOR).** This unordered, binary, presence-based reward gives credit when the student uses the same set of tools as the

teacher, regardless of execution order or call counts:

$$R_a^{\text{set}} = 1\Big[\, S(\tau^{(s)}) = S(\tau^{(t)}) \,\Big].$$

- **Ordered matching.** This variant adds an order constraint by comparing the student and teacher tool sequences position-wise while ignoring tool arguments:

$$R_a^{\text{ord}} = 1\Big[\, \tau^{(s)} = \tau^{(t)} \,\Big].$$

  It tests whether preserving the teacher's tool order is beneficial beyond matching tool choice.

- **Multiset matching.** This variant is order-agnostic but preserves call counts:

$$R_a^{\text{multi}} = 1\Big[\, M(\tau^{(s)}) = M(\tau^{(t)}) \,\Big].$$

  It tests whether matching the number of invocations for each tool improves alignment compared with presence-only matching.

- **F1 (graded set).** This variant replaces binary set equality with a graded overlap score:

$$R_a^{\text{F1}} = \frac{2\,|S(\tau^{(s)}) \cap S(\tau^{(t)})|}{|S(\tau^{(s)})| + |S(\tau^{(t)})|}.$$

  It provides partial credit when the student recovers only part of the teacher's tool set.

- **Recall-only.** This variant rewards coverage of the teacher's tools without penalizing additional tools:

$$R_a^{\text{rec}} = \frac{|S(\tau^{(s)}) \cap S(\tau^{(t)})|}{|S(\tau^{(t)})|}.$$

  It tests whether alignment should focus only on missing teacher tools or also discourage unnecessary extra tools.

All five variants are combined with the same correctness reward R_c and tool-validation reward R_v, and are trained under the same GRPO setup.

Training Dynamics Across Flexible Variants. Figure 10 compares the training dynamics of the five flexible alignment variants with the outcome-only and strict-matching references from Figure 2. The flexible variants show broadly similar behavior: they sustain a high tool-usage rate, unlike the outcome-only baseline, while keeping invalid tool calls substantially lower than strict matching. This pattern suggests that the main trade-off identified in Section 3 can be addressed by a family of flexible teacher-alignment rewards, rather than by set matching alone. In other words, the key mechanism is to provide teacher-guided process feedback that preserves the teacher-selected tool set without enforcing its exact execution order.

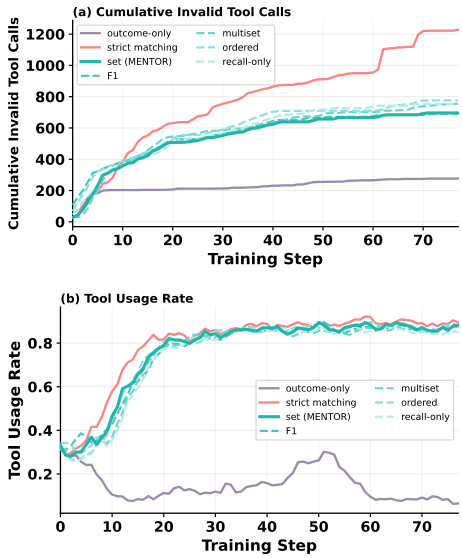

Figure 10: Training dynamics of the five flexible-alignment variants, with the outcome-only and strict-matching references from Figure 2. (a) Cumulative invalid tool calls and (b) tool-usage rate over training steps. The five flexible variants cluster tightly together—all suppressing invalid calls relative to strict matching while sustaining high tool usage, unlike the tool-avoidant outcome-only run—indicating that the learning dynamics are governed by the flexible-alignment principle rather than the specific variant.

Alignment Reward	Math	BFCL	RAG
Ordered matching	26.98	30.21	19.48
Multiset matching	27.21	30.74	20.18
Recall-only	27.05	30.49	19.96
F1 (graded set)	27.43	31.04	20.61
Set matching (MENTOR)	**27.88**	**31.38**	**21.23**

Table 6: Comparison of alternative flexible teacher-alignment rewards on Qwen2.5-7B. All variants are combined with R_c and R_v under the same GRPO setup. Math, BFCL, and RAG are overall task scores. The best score in each column is in bold.

Why We Adopt Set Matching. Although the flexible variants exhibit similar training dynamics, their final task performance differs. As shown in Table 6, set matching achieves the best overall performance across Math, BFCL, and RAG. This supports our choice of set matching as the default alignment reward in MENTOR.

The comparison also clarifies the trade-offs

among alternatives. Ordered matching retains an order constraint, which can penalize valid trajectories that use the teacher's tools in a different executable order. Multiset matching relaxes ordering but requires the student to match the teacher's call counts, which can still penalize benign variation in tool use. Recall-only rewards coverage of the teacher's tools, but does not discourage additional unnecessary tools. F1 provides a graded overlap signal and performs closest to set matching, but its partial-credit structure can weaken the pressure to recover the full teacher tool set. By contrast, set matching focuses on recovering the teacher's core tool-selection behavior while remaining agnostic to order and call counts. This gives the student flexibility in trajectory construction while preserving a clear alignment target.

Implication. These results suggest that the contribution of MENTOR should be understood as a flexible teacher-alignment framework rather than as a method tied exclusively to one particular matching rule. The learning curves in Figure 10 indicate that multiple flexible alignment rewards can address the tool-use trade-off between outcome-only and strict matching. Among these alternatives, Table 6 shows that set matching provides the strongest empirical trade-off in our setting. Other alignment functions may be appropriate in environments where partial overlap, call counts, or ordering information are more important.

C Details for Experimental Setup

C.1 Training Dataset Details

Mathematical Reasoning as a Controlled Testbed. We use mathematical reasoning as the training domain because it provides a controlled testbed for studying reward design in tool-use distillation. Mathematical problems have well-defined answers, structured intermediate computations, and clear feedback from executable tools, allowing us to evaluate final correctness, tool validity, and tool-selection behavior with relatively low ambiguity. This helps isolate the effect of the reward structure from confounding factors such as noisy observations, underspecified goals, retrieval quality, corpus coverage, or environment-specific interaction failures.

We do not train separately on RAG or other domains because our goal is not to conduct an exhaustive cross-domain training study, but to test whether reward learning in a controlled tool-use setting transfers to unseen tool-use scenarios. Retrieval-based tasks introduce additional factors such as document availability, query formulation quality, and evidence retrieval errors, which can obscure whether failures arise from the reward design or from the external environment. We therefore use RAG and BFCL-v4 as out-of-domain evaluation benchmarks rather than additional training domains.

Mathematical reasoning is also sufficiently challenging for capacity-limited SLMs, as solving these problems often requires multi-step reasoning, appropriate tool invocation, and recovery from invalid or unnecessary tool calls. Thus, it provides a useful setting for examining the reward dilemma discussed in Section 3.

Trajectory Construction. To train our models for effective tool use, we construct a high-quality dataset of reference trajectories, as described in Section 5.1. To this end, we use the nvidia/AceReason-Math[5] (Chen et al., 2025b) dataset as our source, leveraging its verified questions and ground-truth answers to ensure reliability.

The construction process involves providing these questions to our teacher model, Qwen3-235B, which generates solution trajectories using a calculator tool. We then filter these outputs, retaining only the successful trajectories in which the teacher's final answer matches the ground truth from the source dataset. This verification process yields a final training set of 1.27k single- and multi-turn trajectories that exemplify successful tool use.

C.2 Versions of Models

We employ Qwen3-235B-Thinking (Qwen3, 2025) as the teacher model and four student models from both Qwen3 (Qwen3, 2025) and Qwen2.5 (Yang et al., 2024) families. The specific model versions and their Hugging Face identifiers are listed in Table 7.

C.3 Tool Execution via Remote Server

All tools described in Tables 8, and 9 are implemented as executable Python code. For tool execution, we use a FastAPI-based remote execution server, following the base architecture of the ReCall framework (Chen et al., 2025a). Our implementation code is publicly available[6].

[5] https://hf.co/datasets/nvidia/AceReason-Math

[6] https://anonymous.4open.science/r/MENTOR-F6E7/

Role	Model	Hugging Face Identifier
Teacher	Qwen3-235B-Thinking	`Qwen/Qwen3-235B-A22B-Thinking-2507-FP8`
Student	Qwen2.5-1.5B-Instruct	`Qwen/Qwen2.5-1.5B-Instruct`
	Qwen2.5-7B-Instruct	`Qwen/Qwen2.5-7B-Instruct`
	Qwen3-1.7B	`Qwen/Qwen3-1.7B`
	Qwen3-8B	`Qwen/Qwen3-8B`

Table 7: Model versions used in experiments

When an agent invokes a tool, the system sends the tool's Python code to the remote server via HTTP API. The server executes the code in a pre-configured environment with necessary libraries installed and returns the result.

C.4 Implementation Details.

The reinforcement learning framework is built on verl (Sheng et al., 2024). The agent is trained for two epochs on the combined training splits of our training dataset. We employ LoRA (Hu et al., 2021) for supervised fine-tuning the student SLMs. For each model, we used the recommended sampling parameters from its official repository. The 'search' tool provided to the agent for the retrieval-based QA tasks is powered by a retrieval environment based on FlashRAG (Jin et al., 2025), using E5-base-v2 (Wang et al., 2022) as the retriever and the Dec. 2018 Wikipedia snapshot (Karpukhin et al., 2020) as the knowledge base. For retrieval-based tasks, we retrieve the top-5 results for each query.

Our training is conducted on 1×8 Nvidia H100 80G GPUs, with full parameter optimization and gradient checkpointing. We provide some important parameter settings in Tables 10 and 11.

C.5 Implementation of the Baseline Models

Outcome-Based RL. To implement the Outcome-Based RL baseline, we adopt the reward structure of ReTool (Feng et al., 2025), which provides reinforcement signals exclusively upon successful final task completion. While the original ReTool framework utilizes Proximal Policy Optimization (PPO), we adapt this approach to the GRPO framework to ensure a fair algorithmic comparison. Under this setup, the reward configuration for the baseline is solely comprised of our correctness reward component (R_c), defined as:

$$R_{\text{c}} = \begin{cases} 1 & \text{if } \hat{y}^{(s)} = \hat{y}^{(t)} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

By removing all process-level feedback, this baseline isolates the student's ability to navigate the tool-use action space when guided only by the final outcome.

Strict RL. To establish the Rigid RL baseline, we adapt the strict trajectory-matching reward mechanism introduced in ToolRL (Qian et al., 2026). While the original ToolRL framework enforces strict sequential matching against curated gold-standard ground-truth trajectories, such explicit human-annotated labels are absent in our distillation setup. Therefore, we adapt the mechanism to use the teacher-generated trajectory $\tau^{(t)}$ as the absolute reference in our GRPO optimization process. Unlike our flexible strategic alignment (R_a), which evaluates tool usage as an unordered set, the rigid reward ($R_{\text{strict format}}$) enforces a strict, position-wise sequential alignment. This requires the student's entire trajectory $\tau^{(s)}$ to exactly match the teacher's reference trajectory step-by-step, encompassing identical tool selection, argument values, and execution order:

$$R_{\text{strict format}} = \begin{cases} 1 & \text{if } \tau^{(s)} = \tau^{(t)} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Any deviation—such as altering the execution order, using a different tool, or generating mismatched arguments—results in a zero reward for that alignment component. This strict formulation intentionally bottlenecks exploratory autonomy, allowing us to accurately isolate the limitations of mechanical trajectory imitation.

C.6 Benchmarks

We evaluate our framework on a set of in-domain and out-of-domain tasks, detailed in Table 12, to measure both task-specific performance and generalization ability. For in-domain tasks, our agent is trained and evaluated on a collection of mathematical reasoning benchmarks that provide a natural gradient of difficulty. This includes the widely-used MATH dataset (Hendrycks et al., 2021)[7], the tool-centric Omni-MATH-512 (Gao et al., 2024)[8], and several Olympiad-level datasets such as AIME24[9], AIME25[10], amc23[11], and minervamath[12]. For out-

[7] https://hf.co/datasets/prithivMLmods/Math-Forge-Hard
[8] https://hf.co/datasets/Heng1999/Omni-MATH-512
[9] https://hf.co/datasets/math-ai/aime24
[10] https://hf.co/datasets/math-ai/aime25
[11] https://hf.co/datasets/math-ai/amc23
[12] https://hf.co/datasets/math-ai/minervamath

Function	Description	Parameter Name	Parameter Description
add	Add two numbers together	firstNumber secondNumber	The first number The second number
subtract	Subtract one number from another	minuend subtrahend	The number to subtract from The number to subtract
multiply	Multiply two numbers together	firstNumber secondNumber	The first number The second number
divide	Divide one number by another	numerator denominator	The number to be divided The number to divide by
sum_numbers	Calculate the sum of an array of numbers	numbers	Array of numbers to sum
floor	Calculate the floor of a number	number	Number to find the floor of
ceil	Calculate the ceil of a number	number	Number to find the ceil of
round_number	Round a number to the nearest integer	number	Number to round
power	Calculate base raised to the power of exponent	base exponent	The base number The exponent
sqrt	Calculate the square root of a number	number	Number to find the square root of
abs_value	Calculate the absolute value of a number	number	Number to find the absolute value of
modulo	Calculate the modulo of two numbers	dividend divisor	The dividend The divisor

Table 8: Math Tool Functions

Function	Description	Parameter Name (Type)	Parameter Description
wikipedia_search	Search Wikipedia for a given query.	query (string) top_n (integer)	Query to search for. Number of results to return. (Optional, default: 5)

Table 9: RAG Tool Function

Parameter	Value
Learning Rate	1e-6
Optimizer	AdamW
Epochs	2
Train Batch Size	16
Mini-batch Size	8
Max Sequence Length	32768
Max Response Length	8192
Number of Rollout	10
Tensor Model Parallel Size	2
Rollout Temperature (Qwen3)	0.6
Rollout Temperature (Qwen2.5)	0.7
GPU Utilization Ratio	0.8
KL Loss Coefficient	0.001
Clip Ratio	0.2
Reward Weights (w_c, w_a, w_v)	0.7, 0.3, 0.1

Table 10: Implementation details of MENTOR.

Parameter	Value
Learning Rate	1e-7
Optimizer	AdamW
Epochs	1
Train Batch Size	4
Gradient Accumulation Steps	16
Weight decay	0.033
Gradient Accum	8

Table 11: Implementation details of SFT.

of-domain tasks, we evaluate the trained agent on tasks requiring tools unseen during training to assess zero-shot generalization. To test *retrieval-based QA*, we reframe multi-hop QA datasets (HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Bamboogle (Press et al., 2023)) as a tool-use problem where the agent is provided with a search(query) tool and must learn to call it effectively. To test broader capabilities, we also use the general *Tool-Calling* benchmark BFCL v4 (Patil et al., 2025), which involves a diverse set of novel tools.

C.7 Teacher-Student Alignment Metric

To quantify alignment in tool usage patterns, we compare the distributions of tool calls across the 12 available tools, which are defined in Table 8. Given the teacher's tool distribution P and a student's distribution Q, we use an alignment score based on the Jensen-Shannon Divergence (JSD), defined as:

$$\text{Alignment}(P, Q) = 1 - \text{JSD}(P, Q)$$
$$\text{where JSD}(P, Q) = \sqrt{\frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M)}$$
$$\text{and } M = \frac{1}{2}(P + Q) \tag{10}$$

The score is bounded between 0 and 1, where 1 signifies a perfect match. D_{KL} denotes the Kullback-Leibler divergence.

Task Type	Dataset Name	Description	Size
Math Reasoning	Math-Forge-Hard	College	500
	Omni-MATH-512 (Gao et al., 2024)	Olympiad	512
	AIME24 (AI-MO, 2024)	Olympiad	30
	AIME25	Olympiad	30
	amc23 (AMC, 2023)	Olympiad	40
	minervamath (Lewkowycz et al., 2022)	College	272
Tool-Calling	BFCL v4 (Patil et al., 2025)	Tool-call	5088
Factual Reasoning	HotPotQA (Yang et al., 2018)	2-hop QA	2000
	2WikiMultiHopQA (Ho et al., 2020)	2-hop QA	2000
	Bamboogle (Press et al., 2023)	2-hop QA	125

Table 12: Benchmarks categorized by in-domain (*Mathematical Reasoning*) and out-of-domain (*Tool-Calling* and *Factual Reasoning*) tasks and their test data size.

Since our objective is tool-use distillation rather than independent policy discovery, teacher alignment serves as a diagnostic measure of capability transfer, not an absolute criterion for policy optimality. Importantly, we use task accuracy and exact match as the primary evaluation metrics, while the alignment score is used to assess whether performance gains are accompanied by successful transfer of tool-use patterns.

Interpretation of Alignment Score. The alignment score is intended as a diagnostic measure for analyzing tool-use transfer in the distillation setting, rather than as an absolute measure of policy optimality. Since our goal is to transfer tool-use capabilities from a stronger teacher model to smaller student models, similarity to the teacher's tool-usage distribution provides a useful behavioral signal for whether the student has acquired teacher-guided tool-selection patterns. However, high alignment alone does not imply that a policy is optimal; therefore, we report AS alongside downstream task performance, using accuracy and exact match as the primary evaluation metrics.

D Supplementary Results and Discussions

D.1 Model Extensibility of Our Framework

Because MENTOR defines alignment in terms of tool-use behavior rather than token-level distributions, its reward design is not tied to a specific tokenizer or model family. We therefore examine whether the same framework can be applied to student models outside the Qwen family. Specifically, we evaluate two LLaMA students using the same Qwen3-235B-Thinking teacher and report the results in Table 13.

Cross-Family Setup. We retain the original Qwen3-235B-Thinking teacher and pair it with two LLaMA students that are close in scale to the stu-

Model	Method	Math	BFCL	RAG
LLaMA-3.2-3B	Vanilla	2.85	22.18	1.92
	SFT	5.62	23.94	2.74
	Outcome-based RL	8.04	25.62	5.81
	Strict RL	8.46	25.43	5.62
	MENTOR	**11.27**	**26.31**	**7.84**
LLaMA-3.1-8B	Vanilla	15.84	23.42	9.86
	SFT	18.06	24.78	10.92
	Outcome-based RL	20.42	28.71	13.74
	Strict RL	20.78	28.92	13.92
	MENTOR	**22.71**	**30.46**	**15.62**

Table 13: Cross-family extensibility of MENTOR. Two LLaMA students are trained with the Qwen3-235B teacher using the two-stage SFT-then-RL pipeline described in the text. The best score in each column is in bold.

dents used in our main experiments: LLaMA-3.2-3B-Instruct and LLaMA-3.1-8B-Instruct. These LLaMA checkpoints are not reasoning-oriented tool-use models and show weaker initial TIR behavior in our sandbox environment. Therefore, directly comparing RL methods from the raw instruction-tuned checkpoints would primarily test whether the model can discover the executable tool-call interface, rather than whether a reward design improves tool-use distillation.

To control for this cold-start confound, we use a two-stage SFT-then-RL protocol for all RL-based methods. Specifically, we first apply a brief SFT warm-up on the same 1.27k filtered teacher trajectories to provide a minimal tool-use initialization, and then apply the method-specific RL objective for Outcome-based RL, Strict RL, and MENTOR. This warm-up is shared across all RL variants and is not specific to MENTOR. The Vanilla and SFT entries follow the same protocol as in the main experiments. For a controlled comparison, we keep the remaining hyperparameters, including rollout count, learning rate, KL coefficient, and reward weights, the same as in the Qwen experiments (Appendix C.4).

Results. Table 13 reports the overall scores on the three benchmark families used in the main paper. For both LLaMA students, MENTOR achieves the best score across all benchmarks. The two RL baselines also improve over SFT in most cases, but remain below MENTOR, mirroring the general ordering observed in the Qwen experiments.

Observations. Three observations can be drawn from Table 13. First, MENTOR consistently outperforms the Vanilla and SFT baselines across both LLaMA scales, suggesting that the proposed reward design can provide useful guidance even when the student model belongs to a different model family than the teacher. Second, MENTOR also outperforms both RL baselines on all three benchmark families, indicating that the gains are not simply due to applying RL after SFT warm-up, but are associated with the flexible, teacher-guided reward structure. Third, the absolute scores of the LLaMA students remain lower than those of the closest Qwen counterparts in Table 1. This is expected, as the LLaMA checkpoints used here are not reasoning-oriented models and show weaker initial TIR behavior in our sandbox environment. These results support our discussion in the Limitations section: initial tool-use ability can facilitate RL exploration, and a brief SFT warm-up provides only a partial initialization rather than a fully optimized TIR training procedure.

Discussion. The goal of this cross-family experiment is not to fully optimize LLaMA-based TIR agents, but to test whether the proposed reward design remains applicable when the student backbone differs from the teacher family. Under the shared SFT warm-up protocol, MENTOR consistently outperforms the SFT-only model and the two RL baselines across the three benchmark families. This indicates that the improvement is not merely due to exposing the LLaMA students to teacher trajectories, but comes from the subsequent flexible reward-based refinement.

At the same time, the absolute scores of the LLaMA students remain below those of the Qwen students in Table 1, suggesting that stronger initial TIR ability still matters. This result is consistent with our limitation discussion: for non-reasoning or weakly grounded backbones, cold-start training is a prerequisite for reliable RL exploration. We leave a more systematic study of cold-start data construction, tool-call format grounding, and model-family-specific TIR initialization for future work.

D.2 Algorithm Extensibility of Our Framework

MENTOR is primarily a reward-design framework: its composite reward combines correctness, teacher alignment, and tool validation over completed rollouts (Equation 3). While our main experiments use GRPO, the reward itself is not specific to GRPO's exact advantage formulation. We therefore examine whether the same reward design remains effective

Algorithm	Reward design	Math	BFCL	RAG
GRPO	Outcome-only	26.13	30.88	18.05
	MENTOR (ours)	**27.88**	**31.38**	**21.23**
DAPO	Outcome-only	25.97	31.02	18.28
	MENTOR (ours)	**27.71**	**31.55**	**21.40**
GDPO	Outcome-only	26.42	31.20	18.41
	MENTOR (ours)	**28.19**	**31.79**	**21.62**

Table 14: Algorithm extensibility of MENTOR. The same composite reward is applied under three policy-optimization algorithms on Qwen2.5-7B: GRPO (critic-free, group-relative), DAPO (a large-scale GRPO variant), and GDPO (a multi-reward-aware optimizer with per-component decoupled normalization). Overall scores follow the conventions of Table 1 and are reported from a single run. The best score within each algorithm group is in bold.

under related group-relative policy optimizers.

Alternative Algorithms. We compare GRPO with two recent variants from the same critic-free, group-relative optimization family. **DAPO** (Yu et al., 2026) modifies the optimization dynamics of GRPO through mechanisms such as decoupled clipping, dynamic sampling, token-level policy-gradient loss, and overlong-reward shaping. **GDPO** (Liu et al., 2026) modifies reward normalization by normalizing each reward component before aggregation, making it particularly relevant for composite rewards such as MENTOR. These two variants allow us to test whether the benefit of MENTOR persists under different optimization dynamics and reward-normalization schemes.

Setup. We re-train Qwen2.5-7B under each optimizer with two reward configurations: the outcome-only reward R_c and the full MENTOR reward $w_c R_c + w_a R_a + w_v R_v$, using the same weights as in the main experiments. All non-algorithm hyperparameters, including learning rate, batch size, KL coefficient, rollout count $G = 10$, training data, and evaluation benchmarks, are kept fixed as reported in Appendix C.4. For DAPO and GDPO, we use their recommended algorithm-specific settings. Table 14 reports single-run results; the GRPO rows correspond to the single-run reward-ablation results in Table 2.

Results. Across all three optimizers, replacing the outcome-only reward with the full MENTOR reward consistently improves Math, BFCL, and RAG performance. The gains are similar across optimizers: $+1.75$ / $+0.50$ / $+3.18$ for GRPO, $+1.74$ / $+0.53$ / $+3.12$ for DAPO, and $+1.77$ / $+0.59$ / $+3.21$ for GDPO on Math, BFCL, and RAG, respectively. This indicates that the benefit of MENTOR is not merely an artifact of the specific GRPO implementation used in the main experiments.

Discussion. These results provide preliminary evidence that MENTOR can be combined with multiple group-relative policy optimizers. The absolute differences among GRPO, DAPO, and GDPO are relatively small, and no single optimizer consistently dominates across all benchmarks. We therefore use GRPO in the main experiments because it offers a simple and competitive critic-free optimization backbone, allowing us to focus the comparison on reward design. Since these results are based on a single run and remain within the group-relative optimizer family, a broader statistically controlled comparison across more policy-optimization algorithms is left for future work.

Scope of Algorithm Comparison. This comparison is not intended to cover all post-training algorithms. We focus on critic-free, group-relative online RL optimizers because they can consume the same rollout-level composite reward used in MENTOR with minimal changes to the training interface. Other algorithms, such as PPO or DPO, introduce additional design factors that would change the scope of the comparison. PPO typically requires a learned value function or critic and additional choices for value estimation and stabilization, while DPO is formulated as a preference-pair objective rather than direct optimization of scalar rollout rewards. Applying MENTOR under DPO would require a separate procedure for constructing chosen–rejected pairs from tool-use trajectories, making it a comparison of training paradigms rather than a controlled test of the reward design. We therefore leave broader comparisons across different post-training paradigms for future work and focus here on whether the proposed reward transfers across related group-relative optimizers.

D.3 Analysis on Computational Cost

RL-based distillation generally requires more training compute than SFT due to repeated rollout generation and policy optimization. The goal of this analysis is therefore not to claim that MENTOR is cheaper to train than SFT, but to clarify the additional cost introduced by our reward design relative to a standard outcome-only RL setup. In particular,

Stage	MENTOR	Standard RL	SFT
Teacher trajectory generation (one-time)	∼1 h	–	∼1 h
RL training	∼6 h	∼6 h	–
SFT training	–	–	∼2 h
Reward computation overhead per step	negligible	negligible	–

Table 15: Wall-clock cost of each training stage on 8× H100 (80 GB). Teacher trajectory generation is shared between MENTOR and SFT (both rely on the same filtered teacher corpus) and is amortized across all subsequent runs. Reward computation in MENTOR consists of set comparisons and binary execution checks, which add negligible per-step overhead relative to the RL rollout itself.

MENTOR adds teacher-guided alignment rewards while keeping the underlying GRPO training procedure unchanged.

Cost Decomposition. Compared with vanilla outcome-only RL, MENTOR introduces two additional cost components. The first is a one-time preprocessing stage in which the teacher model generates reference trajectories used to construct the alignment signal R_a. As described in Appendix B.1, the teacher π_{teacher} (Qwen3-235B-Thinking) is invoked on the AceReason-Math source questions, and the resulting trajectories are filtered against ground-truth answers, yielding a final corpus of 1.27k successful tool-use trajectories.

The second component arises during training, where each student's rollout is scored against the corresponding reference trajectory. This step requires only lightweight operations, such as comparing the tools invoked in the student and teacher trajectories and checking whether tool calls execute successfully. Since these operations are computed outside the model forward and backward passes, their overhead is small relative to the cost of rollout generation and GRPO optimization.

Empirical Wall-Clock Comparison. Table 15 reports the wall-clock cost of each stage on the 8× NVIDIA H100 (80 GB) hardware used in our experiments. MENTOR and Standard RL share the same GRPO backbone with $G = 10$ rollouts per question (Appendix B.1), so their RL training stages have comparable wall-clock costs. The main additional cost of MENTOR comes from the one-time teacher trajectory generation stage, which adds roughly 1 hour in our setup. This suggests that the proposed reward design does not substantially increase the cost of the RL optimization stage itself.

Inference-Time Implications. The practical motivation for distillation is to shift part of the cost from repeated large-model inference to a one-time training process for a smaller student model. MENTOR follows this motivation by improving the tool-use performance of capacity-limited SLMs (1.5B–8B parameters), as shown in Table 1. The actual cost benefit at deployment depends on the number of downstream queries, serving infrastructure, and the relative cost of teacher versus student inference. Nevertheless, the measured overhead indicates that MENTOR incurs only a modest additional training cost compared to standard RL, while enabling the use of smaller distilled models at inference time.

E Details of Instruction Prompts

We utilize prompts for both reference trajectory generation and model evaluation. The specific prompt for reference trajectory generation is shown in Table 17. For evaluation, we assess the robustness of MENTOR and the baseline models using prompts tailored to each of the three domains. All evaluation prompts for the Qwen2.5 and Qwen3 models are created by adapting the official chat templates provided with each model's tokenizer. The prompts are shown in Tables 16, 17, 18, and 19. The instruction prompts used for evaluating the BFCL-v4 benchmark directly follow the format and content specified on the official benchmark website,[13] ensuring consistency with prior studies.[14]

[13] https://gorilla.cs.berkeley.edu/blogs/15_bfcl_v4_web_search.html

[14] https://github.com/ShishirPatil/gorilla/tree/main/berkeley-function-call-leaderboard

Prompt for Qwen2.5 + MATH

SYSTEM:

```
system
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML
    tags:
<tools>
{"type": "function", "function": {"name": "add", ...}}
{"type": "function", "function": {"name": "subtract", ...}}
{"type": "function", "function": {"name": "multiply", ...}}
...
{"type": "function", "function": {"name": "modulo", ...}}
</tools>

For each function call, return a json object with function name and
    arguments within <tool_call></tool_call> XML tags:

<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

USER:

```
Question: Cities $A$ and $B$ are $45$ miles apart. Alicia lives in $A$
     and Beth lives in $B$. Alicia bikes towards $B$ at 18 miles per
    hour. Leaving at the same time, Beth bikes toward $A$ at 12 miles
    per hour. How many miles from City $A$ will they be when they meet
    ?

If you have got the answer, enclose it within \boxed{} with latex
    format.
```

Table 16: Prompt for Qwen2.5 + MATH

Prompt for Qwen3 + MATH

SYSTEM:

```
system
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML
    tags:
<tools>
{"type": "function", "function": {"name": "add", ...}}
{"type": "function", "function": {"name": "subtract", ...}}
{"type": "function", "function": {"name": "multiply", ...}}
...
{"type": "function", "function": {"name": "modulo", ...}}
</tools>

For each function call, return a json object with function name and
    arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

USER:

```
Question: Cities $A$ and $B$ are $45$ miles apart. Alicia lives in $A$
     and Beth lives in $B$. Alicia bikes towards $B$ at 18 miles per
    hour. Leaving at the same time, Beth bikes toward $A$ at 12 miles
    per hour. How many miles from City $A$ will they be when they meet
    ?

If you have got the answer, enclose it within \boxed{} with latex
    format.
```

Table 17: Prompt for Qwen3 + MATH

Prompt for Qwen2.5 + RAG

SYSTEM:

```
system
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML
    tags:
<tools>
{"type":"function", "function":{
  "name":"wikipedia_search",
  "description":"Search Wikipedia for a given query.",
  "parameters":{"type":"object", "properties":{
    "query":{"type":"string", "description":"Query to search for."},
    "top_n":{"type":"integer", "description":"Number of results to
        return. The default value is 5.", "default":5}},
   "required":["query"]}}}
</tools>

For each function call, return a json object with function name and
    arguments within <tool_call></tool_call> XML tags:

<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

USER:

```
Question: What is the capital of France?

If you have got the answer, enclose it within \boxed{} with latex
    format.
```

Table 18: Prompt for Qwen2.5 + RAG

Prompt for Qwen3 + RAG

SYSTEM:

```
system
# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML
    tags:
<tools>
{"type":"function", "function":{
  "name":"wikipedia_search",
  "description":"Search Wikipedia for a given query.",
  "parameters":{"type":"object", "properties":{
    "query":{"type":"string", "description":"Query to search for."},
    "top_n":{"type":"integer", "description":"Number of results to
        return. The default value is 5.", "default":5}},
   "required":["query"]}}}
</tools>

For each function call, return a json object with function name and
    arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

USER:

```
Question: What is the capital of France?

If you have got the answer, enclose it within \boxed{} with latex
    format.
```

Table 19: Prompt for Qwen3 + RAG