MENTOR: A Reinforcement Learning Framework for Model Enhancement via Teacher-Optimized Rewards in Small Models

ChangSu Choi[1]* Hoyun Song[2]* Dongyeon Kim[2] WooHyeon Jung[2] Minkyung Cho[2]
Sunjin Park[3] NohHyeob Bae[3] Seona Yu[3] KyungTae Lim[2]†
[1]Seoul National University of Science and Technology (SEOULTECH)
[2]Korea Advanced Institute of Science and Technology (KAIST)
[3]LG CNS
choics2623@seoultech.ac.kr, {sunjin.park, hyeobiiii, lgyu}@lgcns.com
{hysong, dykim, whitebluej, kveldsstjerne, ktlim}@kaist.ac.kr

Abstract

Distilling the tool-using capabilities of large language models (LLMs) into smaller, more efficient small language models (SLMs) is a key challenge for their practical application. The predominant approach, supervised fine-tuning (SFT), suffers from poor generalization as it trains models to imitate a static set of teacher trajectories rather than learn a robust methodology. While reinforcement learning (RL) offers an alternative, the standard RL using sparse rewards fails to effectively guide SLMs, causing them to struggle with inefficient exploration and adopt suboptimal strategies. To address these distinct challenges, we propose MENTOR, a framework that synergistically combines RL with teacher-guided distillation. Instead of simple imitation, MENTOR employs an RL-based process to learn a more generalizable policy through exploration. In addition, to solve the problem of reward sparsity, it uses a teacher's reference trajectory to construct a dense, composite teacher-guided reward that provides fine-grained guidance. Extensive experiments demonstrate that MENTOR significantly improves the cross-domain generalization and strategic competence of SLMs compared to both SFT and standard sparse-reward RL baselines.

1 Introduction

The augmentation 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 hinder their practical use. This challenge has motivated a line of research focused on smaller language models (SLMs), with the goal of preserving the tool-assisted problem-solving capabilities of larger models (Gao et al., 2023; Gou et al., 2024; Qiu et al., 2025).

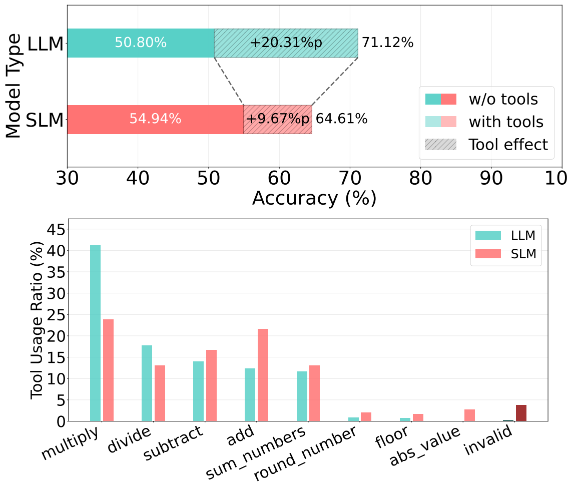

Figure 1: Comparative analysis of the tool effect on model performance and tool invocation patterns between LLMs and SLMs.

Knowledge distillation has been steadily used for transferring such advanced abilities from a larger **teacher** model to a smaller **student** model (Ho et al., 2023; Chenglin et al., 2024; Dai et al., 2024b, 2025; Song et al., 2025; Liao et al., 2025). These studies primarily employ supervised fine-tuning (SFT), where a student model is trained to replicate teacher-generated problem-solving trajectories (Liu et al., 2024; Kang et al., 2025; Lyu et al., 2025). However, the reliance of these methods on a static dataset presents a fundamental scalability issue, as it is impossible to generate trajectories for every scenario the model will face (Luo et al., 2025b; Sun et al., 2025; Yin et al., 2025). As a result, these models often fail to generalize to new tasks, necessitating more adaptive frameworks for teaching tool use.

As an alternative to this imitation-based approach, some studies have explored reinforcement learning (RL) through iterative self-refinement (Trung et al., 2024; Wu et al., 2025b).

*Equally Contributed
†Corresponding Author

To guide the models in learning from their solutions, prior works use simple reward signals, such as correctness of the final answer or tool invocation (Yu et al., 2024; Singh et al., 2025; Wu et al., 2025a). However, this sparse-reward approach is insufficient for SLMs, as their inefficient tool utilization often leads them to adopt suboptimal strategies. For example, Figure 1 demonstrates that LLMs are significantly more effective at leveraging tools, with a tool effect more than double that of SLMs (+20.31% vs. +9.67%). This is linked to suboptimal strategies in SLMs, evidenced by their distinct tool invocation patterns and tendency to make invalid tool calls. This highlights the importance of a dense reward signal to guide the agent toward an effective tool-use strategy.

To address these distinct challenges of both SFT-based distillation and standard RL, we propose a framework that synergistically combines reinforcement learning with teacher-guided distillation. We introduce MENTOR (**M**odel **EN**hancement via **T**eacher-**O**ptimized **R**ewards), a novel approach designed to leverage the strengths of each approach. By employing an **RL-based distillation** process, our framework moves beyond the simple imitation of SFT, allowing the student model to learn a more generalizable policy through exploration. Simultaneously, to address the challenge of reward sparsity that hinders SLMs, we design a **teacher-guided reward**, which utilizes the teacher's trajectory as a reference to construct a composite reward signal. This dense reward provides the fine-grained guidance necessary to steer the SLM towards efficient tool-use strategies and prevent it from converging to suboptimal policies. Our code is publicly available[1].

To summarize, our key contributions are:

- We address the scalability issue of SFT-based distillation by proposing MENTOR, an RL-based distillation framework that learns a more robust and transferable problem-solving methodology through exploration rather than direct imitation.

- To overcome the inefficient exploration and reward sparsity that hinders SLMs in standard RL, we introduce a dense, teacher-guided reward mechanism, providing the fine-grained guidance necessary for learning effective tool-use strategies.

- We demonstrate through extensive experiments that MENTOR significantly improves the cross-domain generalization and strategic competence of SLMs compared to both SFT and standard sparse-reward RL baselines.

2 Related Work

2.1 Tool-Augmented Language Models

A significant line of research enhances the reasoning of LLMs by augmenting them with external tools, such as code interpreters and retrieval APIs. The approaches for integrating these tools are diverse, ranging from prompting-based code generation (Gao et al., 2023; Paranjape et al., 2023; Inaba et al., 2023; Huang et al., 2024) to explicitly training models to invoke tools as part of their reasoning process (Schick et al., 2023; Kong et al., 2023; Qian et al., 2024). Recognizing that effective tool use is an inherently strategic process, recent work has focused on learning optimal tool-invocation patterns through RL (Yu et al., 2024; Feng et al., 2025; Qian et al., 2025). The core challenge in training tool-augmented models is maintaining a robust and generalizable tool-use policy across long reasoning trajectories.

2.2 Reasoning Distillation through SFT

Previous studies have mainly trained student models to clone teacher-generated trajectories as a method for transferring tool-use capabilities to SLMs (Liu et al., 2024; Kang et al., 2025; Lyu et al., 2025). However, SFT-based distillation faces a critical scalability challenge (Sun et al., 2025; Yin et al., 2025), as it is infeasible to curate a static dataset that covers every possible scenario. A key limitation that arises from this data dependency is that models learn to mimic the superficial format of a reasoning trajectory to produce a correct final answer, without internalizing the underlying logical process (Kandpal et al., 2023; Dai et al., 2024a; Li et al., 2025). By contrast, MENTOR employs an **RL-based distillation** framework, where the teacher's trajectory is used not for direct imitation, but serves as a reference to guide an exploratory learning process aimed at developing a more **generalizable** problem-solving method.

2.3 Reinforcement Learning and Reward Design

Reinforcement learning (RL) (Kaelbling et al., 1996) offers a powerful alternative to SFT's

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

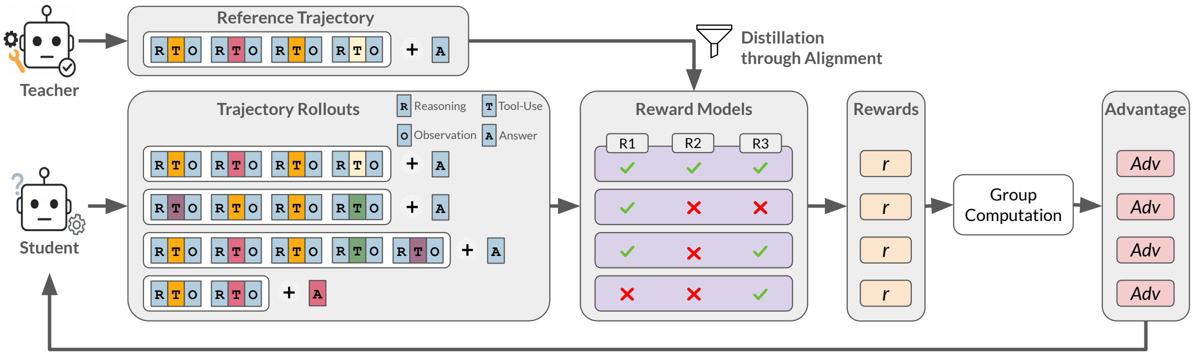

Figure 2: Overview of the MENTOR training framework. A problem-solving trajectory (τ) consists of a sequence of Reasoning (R), Tool-Use (T), and Observation (O), and a final Answer (A). Each student rollout is evaluated by a set of reward models, which generate a reward signal by aligning the student's actions against the teacher's reference trajectory.

imitation-based learning, as its exploratory nature can, in principle, discover more generalizable policies. Foundational algorithms like Proximal Policy Optimization (PPO) established the paradigm of fine-tuning models through policy optimization (Schulman et al., 2017). More recent advancements, such as Group Relative Policy Optimization (GRPO), have adapted this approach for complex reasoning tasks, demonstrating its effectiveness in fostering robust, self-corrective behaviors (Shao et al., 2024).

Many of these RL approaches have been demonstrated on highly capable LLMs for reasoning with tool use. Due to their strong intrinsic abilities, these models can often discover effective policies even when guided by simple, sparse rewards—such as a binary signal for final answer correctness or successful tool invocation (Yu et al., 2024; Feng et al., 2025; Singh et al., 2025; Qian et al., 2025; Wu et al., 2025a). However, SLMs are significantly less efficient explorers (Wei et al., 2022; Xiong et al., 2024). With only sparse rewards, they struggle to link their actions to the final outcome and tend to converge to suboptimal policies. To address this, our work leverages the teacher's trajectory to construct a **teacher-guided reward** signal, providing the fine-grained guidance necessary to steer the SLM's exploration.

3 MENTOR: Model Enhancement via Teacher-Optimized Reward

In this section, we introduce MENTOR, a framework that leverages reinforcement learning (RL) to distill a tool-calling policy from a large teacher model to a smaller student model. The overall process is illustrated in Figure 2. For each input, the teacher model first generates a reference reasoning trajectory that exemplifies a successful problem-solving process. Concurrently, the student model generates multiple exploratory rollouts to sample different reasoning paths. We then apply the Group Relative Policy Optimization (GRPO) algorithm to refine the student's policy. By leveraging a teacher-guided reward signal, MENTOR ensures the student internalizes strategic principles rather than merely memorizing steps. We describe this process in the following subsections.

3.1 Reference Trajectory Generation

First, we use a large teacher model (π_{teacher}) to generate reference reasoning trajectories (τ). Each trajectory consists of a sequence of reasoning (r), tool-use (t), and observation (o) steps. We use 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. For each question x, the teacher model generates a final answer $\hat{y}$ and a reasoning trajectory τ using an instruction prompt (I) as follows:

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

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

3.2 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.

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

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{x_i \sim \mathcal{D},\, \{O_j^{(s)}\}_{j=1}^{G} \sim \pi_{old}(\cdot|x_i)} \left[\frac{1}{G} \sum_{j=1}^{G} \left(\sum_{k=1}^{|\tau_j^{(s)}|} \mathbb{I}(\tau_{j,k}) \right)^{-1} \right.$$
$$\left. \sum_{k=1}^{|\tau_j^{(s)}|} \min\left(\frac{\pi_\theta(\tau_k|\tau_{<k},x_i)}{\pi_{old}(\tau_k|\tau_{<k},x_i)} \hat{A}_{j,k}, clip(\frac{\pi_\theta(\tau_k|\tau_{<k},x_i)}{\pi_{old}(\tau_k|\tau_{<k},x_i)}, 1-\epsilon, 1+\epsilon)\hat{A}_{j,k} \right) - \beta \mathbb{D}_{KL}[\pi_\theta || \pi_{ref}] \right] \tag{3}$$

Algorithm 1 Training with GRPO

```
Require: Student model $\pi_\theta$, old student model $\pi_{\text{old}}$, Teacher model $\pi_{\text{teacher}}$,
    task dataset $\mathcal{D}$, group size $G$, masking function $\mathcal{M}$
for each training iteration do
    for each question $x_i$ do
        Generate reference trajectory $O^{(t)}$ from $\pi_{\text{teacher}}$
        Sample $G$ rollouts $\{O_1^{(s)}, \ldots, O_G^{(s)}\}$ from $\pi_{\text{old}}$
        for each rollout $O_j^{(s)}$ do
            Compute outcome rewards $R(O_j^{(s)}, O^{(t)}))$
        end for
        Compute groupwise advantages $\hat{A}_{j,k}$ for all $O_j^{(s)}$
        Apply loss masking $\mathcal{M}$ to exclude tool output tokens
        Compute GRPO loss $\mathcal{L}_{\text{GRPO}}$ and update $\pi_{\text{student}}$
    end for
end for
```

by the teacher model as described in subsection 3.1. 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 3, 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.

3.3 Teacher-Guided Reward Design

The central challenge of our work lies in the reward design: how to distill a generalizable problem-solving methodology, rather than merely rewarding correct answers. A sparse reward on the final answer is insufficient, as it fails to guide the student towards the teacher's strategic process. To address this, we provide a more fine-grained signal that also evaluates the reasoning process itself by designing a composite reward mechanism with three key components. The total reward for a given student output, $R(O^{(s)}, O^{(t)})$, is defined as:

$$R(O^{(s)}, O^{(t)}) = w_{\text{c}} R_{\text{c}} + w_{\text{a}} R_{\text{a}} + w_{\text{v}} R_{\text{v}}, \tag{4}$$

where w_c, w_a, and w_v are hyperparameters that balance the contribution of each reward component.

Correctness Reward This component evaluates whether the final answer derived by the student model ($\hat{y}^{(s)}$) is consistent with the result produced by the teacher model ($\hat{y}^{(t)}$). This provides a direct signal indicating if the student's overall reasoning process concludes with the same outcome as the teacher's successful demonstration. For our training, we only use reference trajectories where the teacher's answer matches the ground truth (y). The reward is defined as:

$$R_{\text{c}} = \begin{cases} 1 & \text{if } \hat{y}^{(s)} = \hat{y}^{(t)} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Teacher-Alignment Reward To provide an intermediate signal that guides the student towards the teacher's problem-solving strategy, we introduce this reward. This component encourages the student to select the same set of tools as the teacher, which is a crucial aspect of learning the overall methodology. The reward is assigned only if the set of tool calls made in the student's trajectory, $\tau^{(s)}$, is identical to the set of tool calls in the teacher's trajectory, $\tau^{(t)}$. This is defined as:

$$R_{\text{a}} = \begin{cases} 1 & \text{if } \{t_1^{(s)}, \ldots, t_L^{(s)}\} = \{t_1^{(t)}, \ldots, t_L^{(t)}\} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

, where each $t_i^{(s)}$ in $\{t_1^{(s)}, \ldots, t_L^{(s)}\}$ are from $\tau^{(s)}$, and each $t_i^{(t)}$ in $\{t_1^{(t)}, \ldots, t_L^{(t)}\}$ are from $\tau^{(t)}$.

Tool Validation Reward A primary inefficiency we observed (in Figure 1) was the student model's tendency to generate invalid tool calls. These actions, which result in execution errors from the tool interpreter, immediately derail the reasoning process and are a significant source of suboptimal performance for the SLM. To directly penalize this behavior and guide the student towards the valid, error-free trajectories demonstrated by the teacher, we introduce the tool validation reward. This binary reward is 1 only if every tool call within the student's trajectory, $\tau_{student}$, executes successfully without raising an error:

$$R_{\text{v}} = \begin{cases} 1 & \forall o_i^{(s)} \in \tau^{(s)} \text{ is valid} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

4 Experiments and Results

4.1 Experimental Setup

Training Domain Selection We strategically select mathematical reasoning as the primary training domain to enable the model to explore various problem-solving approaches. 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 A.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 capability, we implement a sandbox for running Python code on a remote server. Our implementation code is publicly available. The details of the model versions and tools are in Appendix A.2 and Appendix C, respectively.

Baselines To evaluate our proposed framework, we compare MENTOR against three distinct baselines that represent different training approaches. **1) Vanilla SLM:** The base instruction-tuned model without any further training. This serves as a lower bound for performance. **2) SFT:** This baseline represents the predominant distillation method, where the SLM is fine-tuned on expert-generated trajectories using supervised fine-tuning. **3) Sparse:** To isolate the benefit of our dense reward design, we include a Sparse reward baseline. This agent is trained using the same RL framework as our method, but with a simple reward signal (R_c) based only on the final outcome.

Benchmarks We evaluate our framework on a set of in-domain and out-of-domain tasks, detailed in Table 1, 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), the tool-centric Omni-MATH-512 (Gao et al., 2024), and several Olympiad-level datasets such as AIME24, AIME25, amc23, and minervamath. For **Out-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. We provide details in Appendix A.3.

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

Evaluation Metrics We evaluate the performance on all tasks using Exact Match (EM). To measure the accuracy (Acc), we consider a prediction to be correct only if it exactly matches one of the ground-truth answers after normalization. We also quantify policy alignment using an **alignment score (AS)** based on the Jensen-Shannon divergence. This score measures the divergence between a model's tool usage distribution and the teacher's reference distribution from Figure 1. Further details are in Appendix A.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.

Model	Method	Math (Acc)						
		Math-Forge	Omni-MATH	aime24	aime25	amc23	minervamath	Overall
Qwen 3	235B-Vanilla	66.26	20.31	40.00	26.67	85.00	46.69	52.61
Qwen 2.5	1.5B-Vanilla	5.84	1.37	0.00	0.00	2.50	4.04	2.63
	1.5B-SFT	5.16	1.56	0.00	0.00	5.00	1.47	2.20
	1.5B-Sparse	18.42	3.52	0.00	0.00	**7.50**	**8.46**	6.32
	1.5B-MENTOR	**18.88**	**5.66**	**6.67**	**3.33**	**7.50**	3.31	**7.56**
	7B-Vanilla	35.66	9.96	**13.33**	3.33	40.00	26.47	21.46
	7B-SFT	38.04	11.91	10.00	**6.67**	42.50	25.37	22.75
	7B-Sparse	50.42	13.09	10.00	**6.67**	42.50	29.41	25.35
	7B-MENTOR	**54.06**	**14.65**	6.67	**6.67**	**52.50**	**32.72**	**27.88**
Qwen 3	1.7B-Vanilla	60.10	16.40	26.70	16.70	62.50	**32.00**	35.40
	1.7B-SFT	58.70	16.40	33.30	**23.30**	57.50	28.70	36.60
	1.7B-Sparse	60.00	17.80	26.70	16.70	**70.00**	30.90	36.30
	1.7B-MENTOR	**60.70**	**20.90**	**43.30**	20.00	**70.00**	**32.00**	**41.20**
	8B-Vanilla	65.00	19.50	33.30	23.30	60.00	42.60	40.50
	8B-SFT	65.40	20.31	**36.67**	23.33	62.50	43.38	41.93
	8B-Sparse	**66.49**	22.07	33.33	**36.67**	72.50	43.01	45.68
	8B-MENTOR	65.35	**24.02**	**36.67**	30.00	**77.50**	**43.75**	**46.22**

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	47.64	41.60	38.68	34.60	38.29
Qwen 2.5	1.5B-Vanilla	70.54	**1.88**	**61.29**	2.65	21.49	0.00	3.70	1.60	1.76
	1.5B-SFT	**72.69**	1.37	61.07	2.90	21.62	0.80	3.30	2.60	2.23
	1.5B-Sparse	67.48	1.50	59.22	2.76	21.54	**7.20**	10.50	5.80	7.83
	1.5B-MENTOR	71.48	**2.00**	**61.66**	**3.33**	**22.17**	**7.20**	**11.50**	**6.20**	**8.30**
	7B-Vanilla	71.52	1.25	61.58	3.55	21.88	10.40	13.70	12.60	12.23
	7B-SFT	71.10	2.00	60.47	3.58	21.84	14.40	14.50	12.20	13.70
	7B-Sparse	82.65	**15.25**	72.61	8.49	30.88	20.80	19.00	14.60	18.13
	7B-MENTOR	**82.71**	14.62	**72.83**	**10.36**	**31.38**	**23.20**	**20.10**	**20.40**	**21.23**
Qwen 3	1.7B-Vanilla	82.06	11.50	70.54	4.34	28.74	16.80	20.24	16.57	17.87
	1.7B-SFT	81.06	**11.75**	**70.84**	4.23	28.63	14.40	19.49	17.00	16.96
	1.7B-Sparse	81.54	10.38	**70.84**	4.59	28.46	16.00	19.79	**18.00**	17.93
	1.7B-MENTOR	**82.48**	11.12	70.69	**5.52**	**29.15**	**18.40**	**21.44**	16.80	**18.88**
	8B-Vanilla	87.42	35.38	80.75	9.81	39.25	**35.20**	35.93	30.80	33.98
	8B-SFT	87.83	37.00	80.83	10.57	40.11	**35.20**	37.00	27.60	33.30
	8B-Sparse	88.33	**38.75**	**81.87**	**11.14**	**40.97**	32.00	**38.60**	**31.40**	34.00
	8B-MENTOR	**89.21**	37.88	80.53	**11.14**	40.49	**35.20**	38.00	31.00	**34.70**

Table 2: 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 %). Overall scores are calculated differently: as a macro-average for MATH and RAG, and as an official weighted average for BFCL-v4.

For retrieval-based tasks, we retrieve the top-5 results for each query. We provide further details on hyperparameters in Appendix A.5 and on prompts in Appendix B.

4.2 Experimental Results

The main results, presented in Table 2, demonstrate that our proposed framework, MENTOR, significantly outperforms the baselines.

Distillation Enables Effective Tool Use. On in-domain tasks, both SFT and our RL-based distillation clearly enable more effective tool use, leading to significant performance gains over the vanilla baselines. This confirms that transferring the teacher's tool-calling ability is a broadly successful strategy for improving SLM performance.

RL-Based Distillation Achieves General Performance. On out-of-domain (OOD) benchmarks, the SFT baseline often shows minimal improvement or even performance degradation on OOD tasks, which we attribute to its tendency to overfit on the training domain. By contrast, RL-based approaches consistently yield significant performance gains in the OOD setting. This highlights the effectiveness of the RL-based distillation framework in instilling a more robust and generalizable problem-solving methodology, rather than merely encouraging the memorization of the teacher's trajectories. For a more detailed analysis, the following section focuses on the performance of the Qwen2.5-7B model.

5 Analysis

5.1 Impact of RL-Based Distillation

RL Drives Effective Transfer. To evaluate how efficiently each model learns the teacher's strategy, we analyze the alignment of its tool-use policy with the teacher's patterns. As shown in Figure 3, there is a clear correlation between model performance and alignment score (AS). The SFT model clearly demonstrates imitation failure by learning a policy that is the least similar to the teacher's, con-

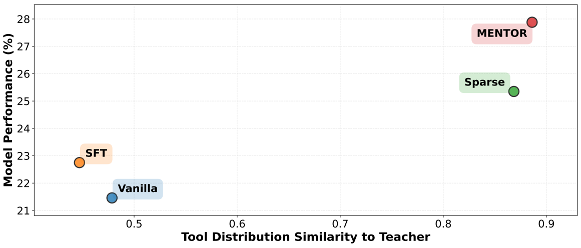

Figure 3: Correlation between task performance (Math) and alignment score (AS).

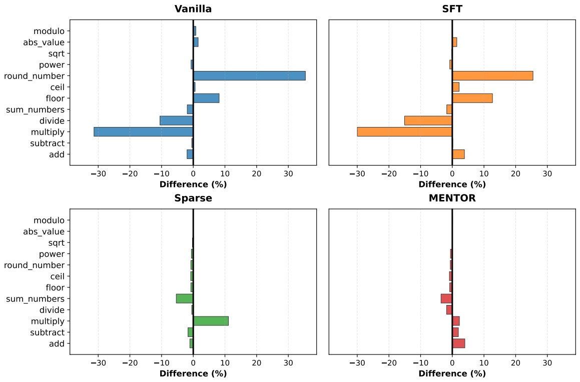

Figure 4: Comparison of tool invocation patterns between student models and the teacher. Each bar represents the percentage point difference in invocation frequency for a specific tool between the student model and the teacher.

firming that it relies on superficial shortcuts rather than the intended methodology. In contrast, the RL-based methods achieve significantly higher performance and alignment with the teacher's policy. Notably, MENTOR achieves the highest accuracy while learning a tool-use policy most similar to the expert teacher's. This suggests that our RL-based distillation is the most effective method for transferring the efficient tool-use strategy.

RL Distillation Drives Policy Alignment. To evaluate the strategic alignment of each model with the teacher, Figure 4 compares their respective tool invocation patterns to the teacher's reference policy, which is shown as the LLM's tool usage in Figure 1. The baseline models, Vanilla and SFT, exhibit significant deviations from the teacher's patterns, indicating that they learn a divergent and suboptimal tool-use policy. By contrast, the RL-based methods, such as Sparse and MENTOR, demonstrate a much closer alignment, with minimal differences across most tools. This provides strong visual evidence that our RL-based distillation is highly effective at teaching the SLM to internalize the teacher's strategic methodology.

RL Enables Efficient Tool Use. Figure 5 shows the tool-use efficiency of each model by plotting the distribution of tool calls per question for both in-domain and out-of-domain tasks. While the Teacher model is highly efficient, the Vanilla and SFT baselines tend to use tools inefficiently, evidenced by their wide distributions and numerous outliers. In contrast, our RL-based approach effectively transfers the teacher's efficient strategy, maintaining a low and stable number of tool calls. It suggests that the RL-based approach successfully transfers the teacher's tool-use efficiency and that this skill generalizes to out-of-domain tasks.

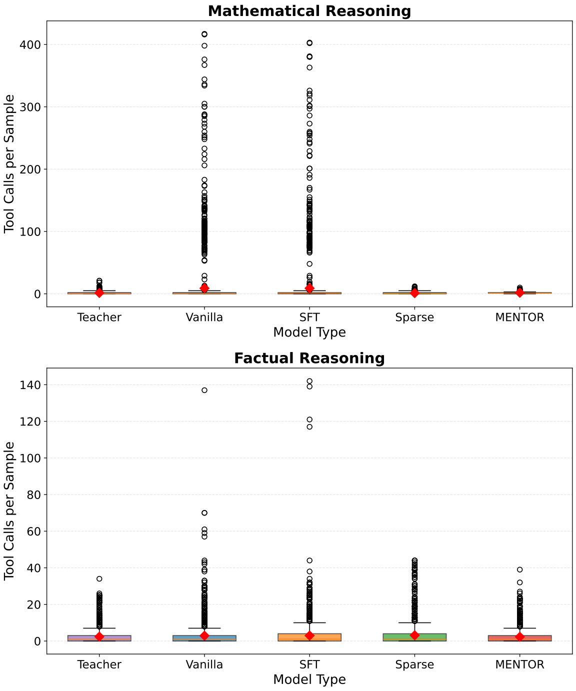

Figure 5: Tool-use efficiency on in-domain and out-of-domain tasks, measured by the distribution of tool calls per sample.

5.2 Impact of Teacher-Guided Reward

Reward Setting	Math	BFCL	RAG	AS
(1) R_{a} (Sparse)	25.35	30.88	18.13	86.84
(2) $R_{\text{a}} + R_{\textbf{Tool format}}$	25.78	30.60	3.58	76.17
(3) $R_{\text{a}} + R_{\text{v}}$	27.80	30.96	8.35	80.94
(4) $R_{\text{a}} + R_{\text{a}}^{\textbf{F1}} + R_{\text{v}}$	26.82	29.53	15.08	83.35
(5) $R_{\text{a}} + R_{\text{a}} + R_{\text{v}}$ (Ours)	**27.88**	**31.38**	**21.23**	**88.79**

Table 3: 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.

Ablation of Rewards We conduct an ablation study to isolate the contribution of each reward component by testing five distinct settings. The results are shown in Table 3. Specifically, **Setting (1)** serves the sparse baseline, using only a reward for final answer correctness (R_{a}). **Setting (2)** adds a reward for the correct tool format, a component previously used in prior works. **Setting (3)** introduces

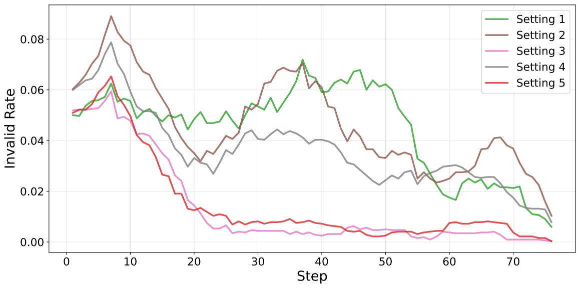

Figure 6: Invalid tool call rate over training steps

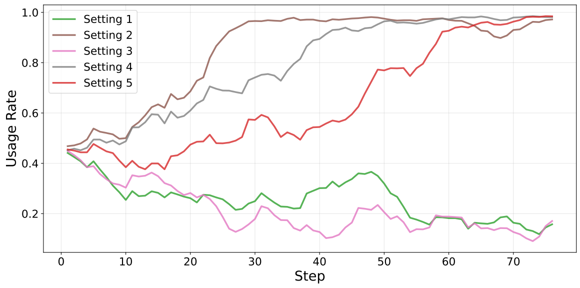

Figure 7: Tool usage rate over training steps

a tool validation reward (R_{v}) to penalize procedural errors. **Setting (4)** evaluates a more flexible teacher-alignment reward by using an F1-score for matching with the teacher's tool set. **Setting (5)** is MENTOR, which combines all rewards for the teacher-guided reward. The results show that our comprehensive reward design (Setting 5) is the best performer, achieving the highest scores across all task benchmarks and in policy alignment.

Kind Guidance is Better than Forcing. The results (Table 3) highlight the necessity of an explicit teacher-alignment reward. A simple reward for invoking any tool (Setting 2) results in poor policy alignment. In contrast, settings that incorporate a teacher-alignment signal (Settings 4 and 5) achieve substantially higher alignment scores and stronger task performance. This suggests that solely encouraging tool use is insufficient, as the model must be guided by the teacher's strategy to facilitate effective learning.

Strict Teacher is Better than Flexible Teacher. Our ablation study also reveals the impact of the alignment reward's strictness. We compare a flexible, F1-based alignment reward (Setting 4) with a strict, exact-match alignment reward (Setting 5). As shown in Table 3, the strict reward in Setting 5 consistently outperforms the flexible reward of Setting 4 across all task benchmarks and achieves a significantly higher policy alignment score. This suggests that providing a clear, unambiguous signal is more effective for guiding the SLM.

Validation Reward Reduces Invocation Errors. To demonstrate the effectiveness of our reward design in addressing tool invocation errors, Figure 6 presents the invalid tool call rate during training for each of the five reward settings from our ablation study. The results show that the choice of reward components leads to distinct learning behaviors. Settings that lack the tool validation reward (R_{v}), such as Settings 1 and 2, fail to effectively reduce their error rates, which remain high and unstable throughout training. In contrast, settings that include the tool validation reward (Settings 3 and 5) learn to avoid invalid calls, with their error rates dropping rapidly to near-zero. This suggests that directly penalizing invalid calls with R_{v} is a highly effective strategy for training a more reliable agent.

Alignment Reward Encourages Tool Use. To analyze how different reward designs affect the model's tendency to use tools, Figure 7 shows the overall tool usage rate for each reward setting during training. A clear divergence in behavior emerges based on whether the reward induces tool use. Models trained without an explicit reward for using tools (Settings 1 and 3) learn to use them less frequently over time, adopting a tool-avoidant policy. In contrast, settings with rewards that encourage tool use (Settings 2, 4, and 5) learn to use tools more frequently. Notably, while Setting 2 (rewarding any tool use) learns this behavior the fastest, our complete reward design (Setting 5) learns more gradually but eventually converges to the same high usage rate. This suggests a more robust learning process, where the correct high-level behavior (frequent tool use) is acquired as a consequence of internalizing the teacher's strategic principles.

6 Conclusion

To address the poor generalization of SFT and the inefficiency of sparse-reward RL in distilling tool-use to SLMs, we introduce MENTOR, a framework that combines reinforcement learning with a dense, teacher-guided reward. Extensive experiments demonstrate that MENTOR significantly improves cross-domain generalization by learning a policy that achieves both closer alignment with the teacher's strategy and superior tool-use efficiency compared to baselines. Our ablation studies confirm that the synergistic combination of each component in our composite reward design is crucial for achieving this robust performance.

Limitations

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

Extending to Other Models Our current framework focuses on the Qwen model series (Qwen2.5 and Qwen3), a choice guided by the design of our RL-based approach. Our method is intended to refine and generalize an existing tool-use policy and thus performs best when the student model has some initial ability. A key open challenge is adapting this framework to models that completely lack this initial ability, as the exploratory feedback from RL alone may be insufficient for them to learn effectively. A promising direction for future work is to explore a two-stage, SFT-then-RL training pipeline. Such an approach could first use an SFT phase to instill a baseline policy before our RL-based refinement is applied, thereby extending the applicability of our method to a broader range of models.

Extending to Real-World Environments Our current work operates within a sandbox environment where tools are invoked via executable code. A significant avenue for future research is to extend our framework to operate in more complex, tool-augmented environments, such as web browsers, simulators, or desktop interfaces. In particular, integration with the Model Context Protocol (MCP) (Anthropic, 2024)—which utilizes servers that can respond to various scenarios—could significantly enhance the capabilities of small agents across a diverse range of real-world tasks. This represents an important 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.

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.

Anthropic. 2024. Introducing the model context protocol. `https://www.anthropic.com/news/model-context-protocol/`.

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*.

Li Chenglin, Qianglong Chen, Liangyue Li, Caiyu Wang, Feng Tao, Yicheng Li, Zulong Chen, and Yin Zhang. 2024. Mixed distillation helps smaller language models reason better. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1673–1690, Miami, Florida, USA. Association for Computational Linguistics.

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

Chengwei Dai, Kun Li, Wei Zhou, and Songlin Hu. 2024b. Improve student's reasoning generalizability through cascading decomposed cots distillation. *arXiv preprint arXiv:2405.19842*.

Chengwei Dai, Kun Li, Wei Zhou, and Songlin Hu. 2025. Capture the key in reasoning to enhance CoT distillation generalization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 441–465. Association for Computational Linguistics.

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*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882.

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.

Tatsuro Inaba, Hirokazu Kiyomaru, Fei Cheng, and Sadao Kurohashi. 2023. MultiTool-CoT: GPT-3 can use multiple external tools with chain of thought prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1522–1532, Toronto, Canada. 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*.

Huanxuan Liao, Shizhu He, Yao Xu, Yuanzhe Zhang, Kang Liu, and Jun Zhao. 2025. Neural-symbolic collaborative distillation: Advancing small language models for complex reasoning tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24567–24575.

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-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

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.

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

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*.

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 Details for Experimental Setup

A.1 Training Dataset Details

To train our models for effective tool use, we construct a high-quality dataset of reference trajectories, as described in Section 3.1. To this end, we use the nvidia/AceReason-Math[2] (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 generated solution trajectories using a calculator tool. We then filter these outputs, retaining only the "successful trajectories" where the teacher's final answer matched 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.

A.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 4.

Role	Model	Hugging Face Identifier
Teacher	Qwen3-235B-Thinking	`Qwen/Qwen3-235B-A22B-Thinking-2507-FP8`
Student	Qwen2.5-1.7B-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 4: Model versions used in experiments

A.3 Benchmarks

We use the MATH[3], Omni-MATH-512[4], AIME24[5], AIME25[6], amc23[7], and minervamath[8] datasets. All datasets are publicly available for research use.

A.4 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 11. 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:

$$\begin{gathered}\text{Alignment}(P, Q) = 1 - \text{JSD}(P, Q) \\ \text{where JSD}(P, Q) = \sqrt{\frac{1}{2}\mathbb{D}_{KL}(P \| M) + \frac{1}{2}\mathbb{D}_{KL}(Q \| M)} \\ \text{and } M = \frac{1}{2}(P + Q)\end{gathered} \tag{8}$$

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

A.5 Hyperparameters for Training and Inference

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 5 and 6.

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

Table 5: 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 6: Implementation details of SFT.

B Details of Instruction Prompts

We utilize prompts for both reference trajectory generation and model evaluation. The specific

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

[3] https://hf.co/datasets/prithivMLmods/Math-Forge-Hard

[4] https://hf.co/datasets/Heng1999/Omni-MATH-512

[5] https://hf.co/datasets/math-ai/aime24

[6] https://hf.co/datasets/math-ai/aime25

[7] https://hf.co/datasets/math-ai/amc23

[8] https://hf.co/datasets/math-ai/minervamath

prompt for reference trajectory generation is shown in Table 8. 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 7, 8, 9, and 10. The instruction prompts used for evaluating the BFCL-v4 benchmark directly follow the format and content specified on the official benchmark website [9], ensuring consistency with prior work [10].

C Tool Execution via Remote Server

All tools described in Tables 11, and 12 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[11].

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 preconfigured environment with necessary libraries installed and returns the result.

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

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

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

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 7: 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 8: 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 9: 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 10: Prompt for Qwen3 + RAG

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 11: 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 12: RAG Tool Function