Distilling LLM Agent into Small Models with Retrieval and Code Tools

Minki Kang[1*] **Jongwon Jeong**[2*] **Seanie Lee**[1] **Jaewoong Cho**[3] **Sung Ju Hwang**[1,4]

[1]KAIST, [2]University of Wisconsin-Madison, [3]KRAFTON, [4]DeepAuto.ai
{minkikang, sjhwang82}@kaist.ac.kr

Abstract

Large language models (LLMs) excel at complex reasoning tasks but remain computationally expensive, limiting their practical deployment. To address this, recent works have focused on distilling reasoning capabilities into smaller language models (sLMs) using chain-of-thought (CoT) traces from teacher LLMs. However, this approach struggles in scenarios requiring rare factual knowledge or precise computation, where sLMs often hallucinate due to limited capability. In this work, we propose **Agent Distillation**, a framework for transferring not only reasoning capability but full task-solving behavior from LLM-based agents into sLMs with retrieval and code tools. We improve agent distillation along two complementary axes: (1) we introduce a prompting method called *first-thought prefix* to enhance the quality of teacher-generated trajectories; and (2) we propose a *self-consistent action generation* for improving test-time robustness of small agents. We evaluate our method on eight reasoning tasks across factual and mathematical domains, covering both in-domain and out-of-domain generalization. Our results show that sLMs as small as 0.5B, 1.5B, 3B parameters can achieve performance competitive with next-tier larger 1.5B, 3B, 7B models fine-tuned using CoT distillation, demonstrating the potential of agent distillation for building practical, tool-using small agents. Our code is available at https://github.com/Nardien/agent-distillation.

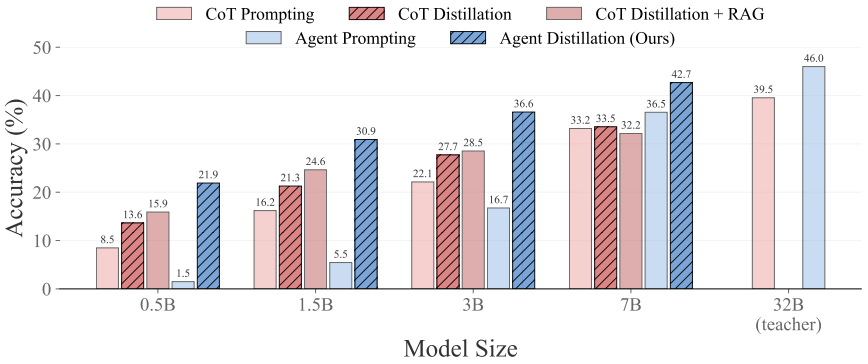

Figure 1: Performance comparison of different sizes of `Qwen2.5-Instruct` models [1] on the average accuracy of four factual reasoning tasks (HotpotQA [2], Bamboogle [3], MuSiQue [4], 2WikiMultiHopQA [5]) and four mathematical reasoning tasks (MATH [6], GSM-Hard [7], AIME [8], OlymMATH [9]). Distillation is done using the 32B model as the teacher and models ranging from 0.5B to 7B as students. Agent distillation consistently improves the performance of smaller models across both domains by enabling them to perform code execution and retrieve information for tasks adaptively. Full results are provided in Table 2.

*Work done at KRAFTON.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

https://doi.org/10.52202/085713-3553

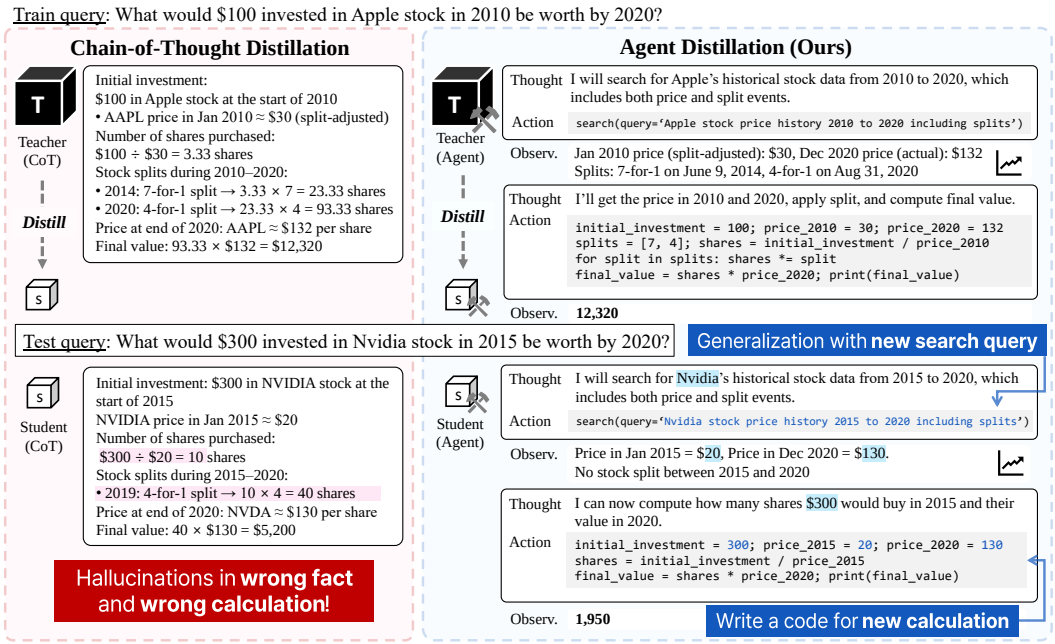

Figure 2: **Concept.** Chain-of-Thought (CoT) distillation trains student models to mimic static reasoning traces from LLMs, but often fails when new knowledge or precise computation is needed at test time. Our proposed agent distillation instead teaches student models to think and *act* (e.g., retrieve facts or execute code) offering stronger generalization and better robustness to hallucination.

1 Introduction

Large language models (LLMs) have achieved remarkable performance across complex real-world tasks, surpassing average human accuracy on college-level mathematics and demonstrating competence in high-stakes domains [10–12]. However, as LLM usage grows, their high inference cost becomes increasingly burdensome. While these considerations have motivated growing interest in smaller language models (sLMs) [13, 14], preserving the problem-solving capabilities of larger models in sLMs remains challenging. Therefore, a core research question emerges: *how can we preserve LLM-level problem-solving ability in much smaller models?*

Although recent advancements in pre- and post-training methods have steadily increased the capabilities of sLMs [15], sLMs still struggle to solve complex tasks at the level of LLMs. To address this, recent works have explored reasoning distillation, where sLMs are trained to mimic CoT reasoning traces generated by teacher LLMs through next-token prediction [10, 11, 1, 16, 17].

However, distilled small models are prone to hallucination and often fail to perform accurate calculations [18]. For example, answering the real-world question, *"What would $100 invested in Apple stock in 2010 be worth by 2020?"*, requires both factual knowledge about stock history and arithmetic reasoning. As illustrated in Figure 2, LLMs can correctly answer this question using CoT by leveraging memorized knowledge and numerical skills. However, simply distilling such a reasoning trace into an sLM does not guarantee generalization, especially those involving new knowledge or calculation not observed during distillation due to their limited capability [19].

In this work, we propose **Agent Distillation**, a framework that moves beyond static reasoning to distill the ability to take action with tools, from LLM agents (*e.g.,* ReAct [20], CodeAct [21]) into sLMs through *reason-act-observe* trajectories. Our goal is to equip sLMs with agentic capabilities: reasoning through problems, taking actions to use code or retrieval tools, observing outcomes, and refining their approach—cloning the behavior of LLM agents. This approach offers two key advantages: (1) sLMs focus on learning how to reason and act to solve problems using tools, rather than memorizing knowledge and calculations, and (2) they generalize better to new queries requiring previously unseen facts or calculations. A remaining challenge is whether such complex agentic behavior can be distilled from a large teacher model (>30B) into a much smaller student (0.5–3B) [1].

https://doi.org/10.52202/085713-3553

To this end, we introduce two simple but effective methods to aid effective distillation. First, we propose a *first-thought prefix* method that aligns agentic reasoning with the teacher model's instruction-tuned behavior, improving trajectory quality of teacher agent without additional fine-tuning. These improved trajectories offer better supervision for sLM distillation. Second, we improve student robustness at test-time through *self-consistent action generation*, which samples multiple trajectories and selects the one yielding a valid and consistent outcome leveraging code interpreter.

We evaluate our agent distillation on four factual (*e.g.,* HotPotQA [2]) and four mathematical (*e.g.,* MATH [6]) reasoning benchmarks. For each reasoning type, we consider one in-domain task and three out-of-domain tasks to test generalization. As in Figure 1, our results show that agent distillation consistently enhances the problem-solving capabilities of small models of 0.5B to 7B.

To summarize, our work makes the following key contributions:

- We propose **Agent Distillation**, a framework for training sLMs to imitate trajectories from LLM agents, enabling agentic behavior without memorizing factual knowledge and calculations.
- We introduce two methods to overcome limitations of naive distillation: (1) a *first-thought prefix* for improving teacher trajectory, and (2) *self-consistent action generation* to boost test-time robustness.
- We validate our method across 8 factual and mathematical reasoning benchmarks, showing strong performance across domains and student model scales (0.5B-7B) compared to CoT distillation.
- Remarkably, we demonstrate that **even 0.5B, 1.5B, and 3B models** distilled with our method can achieve *comparable performance to next-tier larger models* distilled with CoT on average.

2 Related works

2.1 Reasoning distillation of language models

Large language models (LLMs) have shown strong performance on complex reasoning tasks using methods like chain-of-thought (CoT) prompting [22, 23]. To transfer these capabilities to smaller models (sLMs), CoT distillation methods [16, 17, 24–27] train sLMs to reproduce step-by-step reasoning traces from stronger LLMs. This has proven effective—particularly in mathematical reasoning—and is now a common component of post-training pipelines [11, 1]. To improve generalization, recent methods incorporate external tools such as retrieval [19, 28, 29] or code execution [30–32], helping sLMs focus on transferable reasoning strategies rather than memorization of others. Still, most existing approaches rely on static demonstrations and lack interaction with the environment.

In contrast, we distill agentic behaviors where models learn the reasoning and tool use during interactions with environments. This enables sLMs to learn *how to act* for solving tasks.

2.2 Language agents and agentic reasoning

An agent can be broadly defined as an entity that autonomously pursues goals by observing the world and acting upon it. Powered by LLMs, early works like ReAct [20, 33] introduced the concept of *language agents*—which observe the world, *think in natural language*, and act to complete the diverse range of tasks interactively. Since most LLMs are not natively trained for such interaction, prior works have relied on carefully designed prompts (*e.g.,* few-shot examples) for stronger LLMs, and fine-tuned weaker LLMs on trajectories from stronger ones [20, 33–42]. Building on foundations, recent works have pushed language agents toward more advanced agentic capabilities.

Early works focus on teaching LLMs to use tools that enable interaction with external environments [43–47]. Furthermore, agentic retrieval systems have emerged to support multi-hop reasoning over real-world knowledge [48–50], while tool-augmented reasoning leverages external capabilities like code execution to tackle challenging math problems [51–55]. Other approaches promote the notion of *agentic reasoning*, enhancing the decision-making and planning capabilities of LLMs for solving complex tasks with tools through prompting or reinforcement learning [56–58].

Unlike prior work, which primarily focused on fine-tuning LLMs ($\geq$ 7B) on trajectories from stronger close-sourced LLMs (*e.g.,* GPT-4 [12] in FireAct [34]), our work aims to **distill** the agentic capabilities of LLMs into much smaller models (sLMs, $\leq$ 3B), enabling them to operate as capable agents. We address key challenges such as improving the quality of teacher trajectories and optimizing student behavior at test time, building on improved agent framework [21]. We show its effectiveness

https://doi.org/10.52202/085713-3553

across a range of small models (*e.g.,* 0.5B-3B) and tasks requiring strong knowledge and reasoning capabilites–an under-explored yet important setting for practical, small language agents.

3 Preliminary

Knowledge Distillation. Knowledge distillation [59] transfers the capabilities of a large teacher model p_T to a smaller student model p_S. Modern language models follow the auto-regressive transformer architecture [60], where a token-level policy predicts the next token given previous tokens. Given source and target sequences $(\boldsymbol{x}, \boldsymbol{y})$, distillation optimizes the following objective:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{\text{train}}} \frac{1}{L_{\boldsymbol{y}}} \sum_{n=1}^{L_{\boldsymbol{y}}} D(p_T(\cdot \mid \boldsymbol{y}_{<n}, \boldsymbol{x}) \parallel p_S(\cdot \mid \boldsymbol{y}_{<n}, \boldsymbol{x}; \theta)), \tag{1}$$

where D is a divergence metric (*e.g.,* Kullback–Leibler or Jensen–Shannon divergence), and $L_{\boldsymbol{y}}$ denotes the length of the target sequence $\boldsymbol{y}$.

Reasoning distillation. In reasoning tasks, the target sequence $\boldsymbol{y}$ can be a rationale that solves the problem step-by-step. Since collecting human-annotated reasoning is expensive, recent approaches [16, 24, 25, 27] use chain-of-thought (CoT) prompting [23] to generate rationales with large teacher models and train the student to imitate them:

$$\min_{\theta} -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{train}},\boldsymbol{y}\sim p_T(\cdot|\boldsymbol{x},\boldsymbol{I}_{\mathsf{CoT}})} \sum_{n=1}^{L_{\boldsymbol{y}}} \log p_S(\boldsymbol{y}_n \mid \boldsymbol{x}, \boldsymbol{y}_{<n}; \theta), \tag{2}$$

where $\boldsymbol{I}_{\mathsf{CoT}}$ denotes a CoT-style prompt such as "Let's think step by step." [23].

4 Agent Distillation

While reasoning distillation is effective and has become a standard post-training technique [11, 1], it does not equip models with the ability to interact with external environments through actions. Recent work [20, 21] shows that large models can generate actions grounded in intermediate reasoning, observe feedback from the environment, and adapt accordingly.

We refer to such interactive sequences as *agent trajectories*, consisting of repeated cycles of thought ($\boldsymbol{r}$), action ($\boldsymbol{a}$), and observation ($\boldsymbol{o}$). Given an input $\boldsymbol{x}$, the teacher model generates a trajectory:

$$\tau = ((\boldsymbol{r}_1, \boldsymbol{a}_1, \boldsymbol{o}_1), \ldots, (\boldsymbol{r}_{L_\tau}, \boldsymbol{a}_{L_\tau}, \boldsymbol{o}_{L_\tau})) \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{I}_{\mathsf{agent}}), \tag{3}$$

where $\boldsymbol{I}_{\mathsf{agent}}$ is an instruction prompt for the agent (*e.g.,* "To solve the task, you must plan forward to proceed in a series of steps, in a cycle of Thought:, Code:, and Observation: sequences" [21, 61]). Each observation $\boldsymbol{o}$ comes from the environment in response to action $\boldsymbol{a}$, not generated by the model.

Following prior works [34, 52], we fine-tune the student model on generated trajectories, excluding observations from the loss:

$$\min_{\theta} -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{train}},\tau\sim\pi_T(\cdot|\boldsymbol{x},\boldsymbol{I}_{\mathsf{agent}})} \sum_{t=1}^{L_\tau} \log p_S(\boldsymbol{r}_t, \boldsymbol{a}_t \mid \boldsymbol{x}, \tau_{<t}; \theta), \tag{4}$$

where $\tau_{<t} = ((\boldsymbol{r}_1, \boldsymbol{a}_1, \boldsymbol{o}_1), \ldots, (\boldsymbol{r}_{t-1}, \boldsymbol{a}_{t-1}, \boldsymbol{o}_{t-1}))$.

This distillation enables student models to function as interactive agents. For instance, a model distilled from CodeAct [21] can reason about which code snippet to generate, generate actions as codes (*e.g.,* API calls, loops), and respond to execution feedback. If the interpreter returns an error, the model can revise the code accordingly; if the output is valid but insufficient (*e.g.,* suboptimal search results), it can rephrase the query and continue the task adaptively.

Despite its promise, agent distillation presents two key challenges, particularly when applied to small language models (sLMs). First, agentic behavior often lies out-of-distribution relative to the pre-training and instruction-tuning distribution of both teacher and student models. As a result, distilling such behavior may degrade performance on domains where the student is already well-optimized for CoT-style reasoning. Second, although sLMs are pretrained on large code corpora [62], they may struggle to produce functional code during inference. Typical failure cases include misformatted code outputs or incorrect usage of library functions, which hinder the ability of agents to interact.

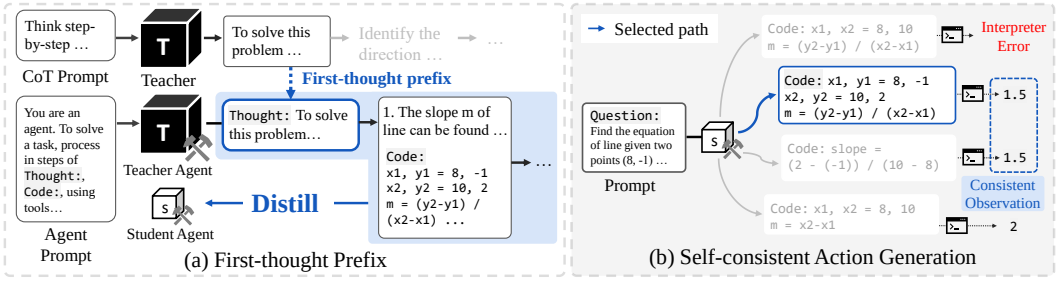

Figure 3: **(a) First-thought Prefix:** We prompt teacher with a CoT prompt to induce step-by-step reasoning. The first reasoning step is used as a prefix to generate an agentic trajectory, which is then distilled to a student agent to teach CoT-style reasoning initialization. **(b) Self-consistent Action Generation:** The agent generates multiple candidate actions and selects the one with consistent outcomes. Thoughts are omitted for brevity.

First-thought prefix. We observe that instruction-tuned LLMs (*e.g.,* Qwen2.5-32B-Instruct [1]), when employed as agents, demonstrate reduced performance on challenging problems from MATH500 benchmarks compared to their performance with CoT prompting [23] (see Appendix D.1 for experimental results). This degradation can further propagate during distillation, negatively impacting student models where they have also been instruction-tuned on CoT-style data.

We hypothesize that instruction-tuned models, which have already been trained to produce CoT reasoning to solve the task, can exhibit distributional drift when prompted with agent instructions (*e.g.,* Prompt D.2 in Appendix). Although these models are capable of structured reasoning, the additional instruction to generate reason-act trajectories may override or conflict with their original reasoning patterns. As a result, the model may deviate from the correct reasoning path it would otherwise follow under CoT prompting. Since prior studies have shown that the initial reasoning step critically determines the final conclusion of LLMs [63–65], ensuring that the model begins reasoning in an appropriate direction during its first action generation becomes essential for maintaining accurate reasoning.

To this end, we propose the first-thought prefix (FTP). Motivated by the prefix-attack in LLM jailbreaking works [66–68], this method integrates the initial reasoning step from a CoT prompting as a prefix to the agent's first thought as in Figure 3(a). Formally, we modify the trajectory sampling described in Equation 3 as follows:

$$\boldsymbol{y}_1 \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{I}_{\texttt{CoT}}), \quad \tau = \{(\boldsymbol{r}'_1, \boldsymbol{a}_1, \boldsymbol{o}_1), \ldots, (\boldsymbol{r}_{L_\tau}, \boldsymbol{a}_{L_\tau}, \boldsymbol{o}_{L_\tau})\} \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{I}_{\texttt{agent}}), \tag{5}$$

where $\boldsymbol{y}_1$ is the first-step of CoT reasoning and $\boldsymbol{r}'_1$ denotes the completed first thought of the agent following the prefixed first-step $\boldsymbol{y}_1$. Note that this method is **only used to generate trajectories from the teacher agent**; the student agent does not explicitly require first-thought prefix during inference.

Self-consistent action generation. We observe that small distilled agents often produce invalid actions, particularly in the context of CodeAct [21], where invalid actions refer to code that either fails to execute or throws errors. To improve robustness in action generation, we introduce self-consistent action generation (SAG). Instead of using greedy decoding, we sample multiple N thought-action sequences for each step through nucleus sampling [69] with a high temperature to encourage diversity. We then filter out any sequences that result in parsing or execution errors using a lightweight code interpreter. When all generated actions fail, we retain one randomly selected failed action and feed its error message back as an observation, allowing the model to self-correct in subsequent steps. To further ensure correctness, we perform majority voting over the resulting observations [70], selecting the action whose output is most consistent across samples. For example, in Figure 3(b), the agent generates four candidate sequences. One result in an interpreter error is filtered out. Among the remaining three, two produce the same output, so we select one of these two consistent actions as a final action.

5 Experimental setup

We evaluate our proposed Agent Distillation across benchmarks to test whether small language models (sLMs) can acquire agentic abilities from a large language model (LLM) agent teacher.

https://doi.org/10.52202/085713-3553

Table 1: Task categorization with domain and sampled test data size we used.

Task Type	Domain	Dataset Name	Description	Test Data Size
Factual Reasoning	In-domain	HotPotQA [2]	2-hop question-answering	500
	Out-of-domain	Bamboogle [3]	2-hop question-answering	125
	Out-of-domain	MuSiQue [4]	3-hop question-answering	500
	Out-of-domain	2WikiMultiHopQA [5]	2-hop question-answering	500
Math Reasoning	In-domain	MATH [6]	College-level math	500
	Out-of-domain	GSM-Hard [7]	Large number arithmetics	500
	Out-of-domain	AIME [8]	Olympiad-level problems	90
	Out-of-domain	OlymMath [9]	Olympiad-level problems	200

Tasks and datasets. We evaluate two categories of reasoning tasks: factual and mathematical. For each, we assess both in-domain and out-of-domain generalization. We use 1,000 HotPotQA [2] and 2,000 MATH [6] examples for training. For test benchmarks, we summarize them in Table 1. To reduce evaluation cost, we limit each test set to 500 examples, following Wang et al. [71]. As a metric, we use exact match for math and llm-as-a-judge [72] using `gpt-4o-mini` for factual reasoning.

Models. The teacher model is `Qwen2.5-32B-Instruct`, a 32B parameter instruction-tuned model. For student models, we use the `Qwen2.5-Instruct` series with four sizes: 0.5B, 1.5B, 3B, and 7B parameters. All student models are instruction-tuned prior to distillation [1].

Baselines. We compare two main distillation paradigms: (1) CoT distillation [16], which transfers static reasoning traces generated using Chain-of-Thought prompting, and (2) our proposed Agent Distillation, which transfers interactive reason-act-observe trajectories. For CoT distillation, we add the baseline that uses retrieval-augmented generation [73] in both distillation and inference for a fair comparison with external knowledge [19, 28, 29]. For ours, we adopt the formulation from CodeAct [21, 61], where each step consists of a `Thought`, `Action` (*e.g.,* Python code), and `Observation`. Additionally, we incorporate two proposed methods — distillation using trajectories through first-thought prefix `FTP` and self-consistent action generation `SAG`.

Training & inference details. For reproducibility of experiments, we use Wikipedia 2018 as a knowledge base for both agents and RAG instead of search engine. We use `e5-base-v2` [74] as both document and query embeddings as in Jin et al. [49]. For both CoT and agent, we sample one trajectory per question from the teacher model and **filter out wrong trajectories**, resulting in approximately 2,000 trajectories for distillation.

We fine-tune student models using parameter-efficient tuning with LoRA (rank 64) on all linear layers [75]. All models are fine-tuned for 2 epochs using a batch size of 8 and a learning rate of $2 \cdot 10^{-4}$. All experiments are conducted using four NVIDIA A100 80GB GPUs.

For inference, we use a greedy decoding. For all agents, we set max steps to 5. For `SAG` in main experiments, we set $N = 8$ with temperature to 0.4. More details are in Appendix C.

6 Results

Overall results. In Table 2, we find that agent distillation consistently improves performance across all model sizes. Before distillation, most sizes of models (except 7B) fail to produce effective agentic outputs via prompting alone, often generating incorrect or unparseable code action. In contrast, our distilled agents outperform CoT-distilled counterparts, particularly on out-of-domain tasks across both factual and mathematical domains. These results highlight the **effectiveness of agent distillation in improving generalization of sLMs**. Notably, the gains are further amplified by our two methods–First-thought Prefix (`FTP`) and Self-consistent Action Generation (`SAG`).

Our findings also demonstrate that **agent distillation enables small models to match or exceed the performance of CoT-distilled models that are 2–4× larger**, offering a promising path toward efficient and capable language agents. Specifically, the 0.5B agent matches the performance of a 1.5B CoT-distilled model, the 1.5B agent reaches its 3B counterpart, the 3B agent surpasses the 7B CoT model, and the 7B agent even outperforms the 32B CoT model.

https://doi.org/10.52202/085713-3553

Table 2: **Main results.** Distilled agents show the strong performance on most of tasks, especially on out-of-domain tasks, compared to baselines. FTP = First-Thought Prefix, SAG = Self-consistent Action Generation. Highlighting best among same-sized models. Avg. denotes the average score across all tasks.

Params		Method	In-domain		Out-of-domain						Avg.
			HotPot QA	MATH 500	MuSi-Que	Bamb-oogle	2Wiki QA	GSM-Hard	AIME	Olym-MATH	
Teacher: Qwen-2.5-Instruct											
32B	CoT	Prompting	36.8	79.2	12.2	60.8	33.4	74.6	13.3	6.0	39.54
	Agent	Prompting	56.4	69.2	25.2	58.4	49.8	76.4	21.1	11.5	46.00
Student: Qwen-2.5-Instruct											
7B	CoT	Prompting	29.2	71.8	5.8	43.2	29.2	66.6	12.2	7.5	33.19
		Distill	31.0	72.6	9.0	44.8	26.8	67.6	10.0	6.5	33.54
		Distill + RAG	42.8	68.0	6.6	40.0	27.6	60.6	6.7	5.0	32.16
	Agent	Prompting	46.8	56.0	16.8	41.6	45.6	62.2	13.3	10.0	36.54
		Distill	51.2	62.2	19.6	52.0	45.2	72.0	11.1	5.5	39.85
		+ FTP	55.0	66.6	17.6	56.0	44.6	70.8	14.4	13.0	42.26
		+ SAG	53.2	64.0	20.6	50.4	48.2	73.4	15.6	9.5	41.86
		+ FTPSAG	54.4	67.8	19.4	55.2	45.2	72.4	15.6	11.5	**42.68**
3B	CoT	Prompting	38.6	62.8	6.2	33.6	21.6	60.2	6.7	4.5	29.27
		Distill	26.8	61.8	6.4	34.4	25.0	56.8	5.6	5.0	27.72
		Distill + RAG	40.6	59.6	4.6	32.0	28.2	53.2	5.6	4.5	28.53
	Agent	Prompting	38.6	30.5	8.8	29.6	28.8	25.8	4.4	3.0	21.20
		Distill (Ours)	48.4	54.0	13.0	37.6	37.4	64.2	6.7	7.5	33.60
		+ FTP	47.6	54.4	13.0	43.2	41.4	63.0	7.8	5.5	34.49
		+ SAG	48.6	57.4	13.0	36.0	37.4	65.6	0.0	10.0	33.50
		+ FTPSAG	49.4	60.2	15.8	38.4	41.0	65.4	15.6	7.0	**36.60**
1.5B	CoT	Prompting	17.8	47.6	3.0	21.6	19.0	49.0	1.1	3.5	20.33
		Distill	23.8	46.4	2.0	21.6	18.4	51.0	5.6	1.5	21.28
		Distill + RAG	37.6	48.6	4.2	26.4	27.0	48.6	2.2	2.5	24.64
	Agent	Prompting	8.6	22.2	1.6	10.4	10.6	9.0	1.1	0.0	7.94
		Distill (Ours)	43.0	46.8	9.0	27.2	35.6	54.8	1.1	7.0	28.06
		+ FTP	43.6	46.4	8.0	30.4	32.6	60.6	7.8	3.5	29.11
		+ SAG	43.8	49.8	11.6	31.2	36.6	58.0	7.8	3.5	30.29
		+ FTPSAG	45.6	50.6	9.2	33.6	33.6	60.6	6.7	4.5	**30.55**
0.5B	CoT	Prompting	9.2	28.4	0.2	7.2	12.8	25.6	1.1	4.0	11.06
		Distill	13.2	28.6	1.4	10.4	23.8	28.6	1.1	2.0	13.64
		Distill + RAG	29.2	28.0	1.6	13.6	25.4	27.4	0.0	2.0	15.90
	Agent	Prompting	2.4	3.0	0.0	0.8	2.8	5.4	0.0	0.0	1.80
		Distill (Ours)	34.6	30.4	7.0	17.6	28.8	31.2	3.3	1.0	19.24
		+ FTP	32.4	28.8	3.4	24.0	30.8	36.4	1.1	3.0	19.99
		+ SAG	34.0	33.8	8.2	13.6	33.0	33.0	4.4	0.0	20.01
		+ FTPSAG	33.4	34.4	5.6	24.0	31.2	40.8	3.3	2.5	**21.90**

Table 3: Comparison of performance across general and code-specific models. 32B/1.5B denote general models and 32B-Coder/1.5B-Coder denote code-specific models. For all models, we apply SAG with $N = 8$.

Teacher	Student	HotPot QA	MATH 500	MuSi-Que	Bamb-oogle	2Wiki QA	GSM-Hard	AIME	Olym-MATH	Avg.
32B	1.5B	45.6	50.6	9.2	33.6	33.6	60.6	6.7	4.5	30.55
32B-Coder	1.5B	42.6	51.4	10.0	36.8	36.8	60.0	6.7	3.0	**30.91**
32B	1.5B-Coder	37.8	52.6	8.2	30.4	38.0	59.8	3.3	6.0	29.52
32B-Coder	1.5B-Coder	41.4	49.2	9.4	30.4	37.4	63.6	4.4	5.5	30.17

Factual reasoning results. We find that retrieval improves the performance of CoT-distilled models on factual reasoning benchmarks. However, due to its static nature, it can degrade performance on tasks requiring dynamic or adaptive information use, such as mathematical reasoning. In contrast, our distilled agents outperform even RAG-enhanced CoT models. This is because agent distillation equips the model to actively retrieve and integrate knowledge during reasoning, rather than relying solely on pre-fetched documents that may be insufficient or misaligned with the task.

Math reasoning results. On mathematical reasoning tasks, our distilled agents demonstrate strong overall performance. The 1.5B, 3B, and 7B models show improvements on the AIME and Olym-

https://doi.org/10.52202/085713-3553

Table 4: Comparison of performance across `Llama-3.2-1B-Instruct` [11] and `Phi-4-mini-instruct` [76] models. Teacher model is `Qwen-2.5-32B-Instruct`. Performance trends are consistent to results in Table 2.

Student Model		Method	HotPot QA	MATH 500	MuSi-Que	Bamb-oogle	2Wiki QA	GSM-Hard	AIME	Olym-MATH	Avg.
Llama-3.2-1B -Instruct	CoT	Prompting	13.2	28.8	1.2	14.4	8.0	19.0	1.1	2.5	11.53
		Distill	18.2	25.6	2.6	25.6	19.0	13.8	1.1	2.0	13.23
	Agent	FT	36.0	34.6	2.6	11.2	26.4	40.4	1.1	2.0	19.54
		+ FTP	37.6	32.8	3.6	24.0	30.8	45.0	1.1	1.5	22.93
		+ FTPSAG	40.6	40.0	3.2	23.2	30.0	47.8	1.1	3.0	**23.97**
Phi-4-mini -instruct (3.8B)	CoT	Prompting	24.2	53.8	6.0	38.4	24.2	49.6	5.6	4.5	25.04
		Distill	24.4	63.2	5.8	33.6	24.8	54.8	6.7	7.0	27.41
	Agent	Distill	48.2	52.4	8.8	27.2	33.6	69.4	5.6	6.0	31.52
		+ FTP	45.2	60.0	7.2	34.4	39.2	71.2	10.0	7.5	34.58
		+ FTPSAG	47.0	65.6	9.6	32.0	41.0	73.0	11.1	7.0	**35.79**

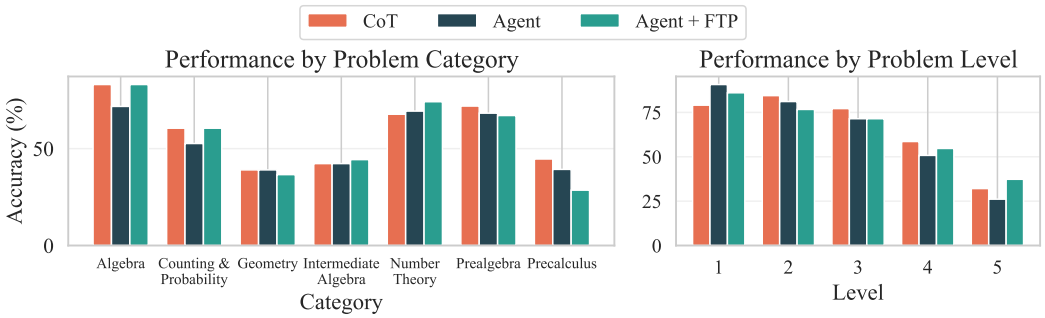

Figure 4: **Performance comparison on the MATH subcategories and levels** between CoT and Agent distillation of 3B models. Left: Accuracy by problem category. Right: Accuracy by problem difficulty level. The results highlight that FTP improves the performance of small agents in harder problems.

MATH benchmarks, benefiting from code tool use for complex calculations acquired through distillation. On GSM-hard, agent distillation improves robustness in reasoning over rare number combinations, such as 6-digits arithmetic. While performance on MATH500 lags behind CoT-distilled models in 3B and 7B models, we attribute this to the Qwen2.5 series being heavily instruction-tuned on college-level math, which may align better with CoT. Furthermore, we conjecture that larger models (3B and 7B) possess stronger internal computation skills, making tool use less beneficial on benchmarks like MATH500, while smaller models benefit more from external code execution. Nonetheless, agent distillation remains effective for larger models on harder math problems (e.g., GSM-Hard, OlymMATH), where the agentic method yields consistent gains. Overall, agent distillation delivers substantial gains across a wide range of math tasks. We provide a detailed breakdown in Section 7.

7 Analysis

Code-specific teacher yields better students—marginally. We primarily study general instruction-tuned models for both the teacher and student agents, as shown in Table 2. Given that CodeAct [21] requires generating code to perform actions, a natural question arises: Can we obtain better agents by using code-specific models for the teacher or student in the agent distillation process?

To explore this, we conduct the same set of experiments using `Qwen2.5-Coder-32B-Instruct` as the teacher and `Qwen2.5-Coder-1.5B-Instruct` as the student [77]. The results, presented in Table 3, suggest that the use of a code-specific student model does not significantly impact performance. Instead, the choice of a code-specific model as the teacher appears to be more influential in generating effective trajectories for distillation. Nevertheless, the overall improvements are marginal on average, indicating that code-specific post-training has limited impact, which suggests the code knowledge is not critical bottleneck of the student.

Agent distillation applies across different model families. We further validate whether the improvements from our method generalize across different language model families. In Table 4, we conduct experiments with two additional student models, `Llama-3.2-1B-Instruct` [11] and `Phi-4-mini-instruct` [76]. Results show that both models benefit from agent distillation com-

Figure 5: Comparison of SAG in agents and **self-consistency** [70] **in CoT** for 3B models: self-consistency in CoT is helpful in math tasks but not in factual tasks.

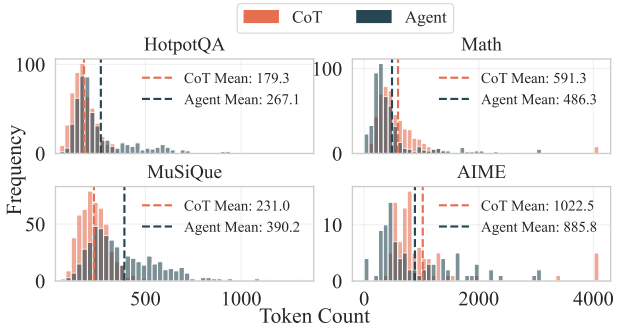

Figure 6: **Generated token counts** comparisons in 3B models. For factual reasoning tasks (HotpotQA, MuSiQue), agent generates more tokens than CoT. In contrast, for math reasoning tasks (MATH, AIME), CoT generates slightly more tokens than agent.

pared to CoT distillation. Moreover, both FTP and SAG yield consistent improvements across the two models, demonstrating that our proposed method is broadly applicable to different model families.

First-thought prefix improves the agents on more complex reasoning problems. In Table 2, we observe that agent distillation does not improve performance on MATH500 compared to CoT distillation, particularly for the 3B model. To investigate further, we break down MATH500 performance by both problem category and difficulty level.

Interestingly, naive distillation degrades the performance of distilled 3B agent on most of levels. However, when using teacher trajectories with a first-thought prefix, distilled 3B agent shows improved performance on level 4 and 5 problems–**with especially significant gains at level 5.** These results suggest that trajectories from FTP help student agents become more robust on complex reasoning tasks, a trend also observed in the challenging AIME benchmark in Table 2.

However, a remaining concern is the performance drop in certain categories—most notably, a decline in precalculus. Our analysis suggests that this degradation is primarily due to the nature of certain problem types that require an analytic approach rather than straightforward calculations (*e.g.,* applying properties of trigonometric functions). Such problems are harder to solve using code tools. We explore this issue in detail in Appendix D.

Self-consistency improves CoT, but the agent with SAG still performs better. Self-consistent action generation (SAG) enhances small agents by filtering out invalid code actions and retaining only those that are consistent with observations. Similarly, self-consistency [70] can be applied at test time in Chain-of-Thought (CoT) reasoning to improve performance without relying on an external verifier.

A natural question is whether CoT with self-consistency, using the same computational budget, can outperform an agent with SAG. To investigate this, we conduct experiments using self-consistency [70] on CoT-distilled small language models (sLMs), applying majority voting over multiple samples.

As shown in Figure 5, in the MATH benchmark—where CoT already surpasses the agent with SAG—self-consistency further improves the performance of the CoT-distilled model. However, in the more challenging AIME benchmark, the small agent with SAG still outperforms the CoT-distilled model under the same generation budget. Moreover, in factual reasoning tasks such as HotpotQA and MuSiQue, self-consistency yields only marginal gains, suggesting limited effectiveness in these settings.

How many tokens should agents generate? A natural question is whether a distilled agent should generate significantly more tokens than a CoT-distilled model, potentially affecting the efficiency and practicality of small models. To investigate this, we analyze token counts on two factual and two math reasoning tasks using 3B distilled models.

As shown in Figure 6, there is no significant difference in total token generation between the two approaches across both domains. In factual reasoning, the agent tends to generate more tokens due to making multiple retrieval calls across several steps to gather accurate information. In contrast, in math reasoning, the agent generates fewer tokens than CoT models by delegating repetitive calculations to code execution, often leveraging logical structures like for-loops.

https://doi.org/10.52202/085713-3553

Figure 7: **Impact of Self-consistent Action Generation** (SAG) on code generation errors across models and 3 math datasets. SAG consistently reduces code parse (dark) and code execution (light) errors, especially for smaller models (0.5B) and the AIME dataset.

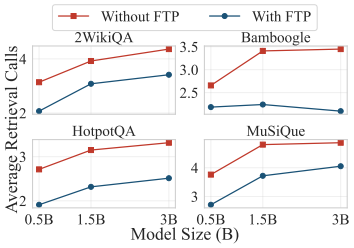

Figure 8: **Average retrieval tool calls** across three model sizes and datasets. Harder tasks and larger sizes make agents use more retrieval calls.

SAG significantly reduces invalid code actions. In Figure 7, we show the effect of self-consistent action generation (SAG). SAG reduces the generation of codes with both parsing and code execution errors. This result indicates that the small distilled agent is capable of generating valid code but the likelihood of generating valid code tends to decrease with smaller model sizes. SAG mitigates this issue by sampling multiple actions per turn, increasing the likelihood of generating a valid one. Nevertheless, execution errors may still occur, in which case the agent uses the error message as feedback to revise its code in the next turn.

Larger agents make more retrieval calls, FTP reduces them We analyze how frequently agents use the retrieval tool across different model sizes and factual reasoning benchmarks. As shown in Figure 8, larger models tend to make more retrieval calls than smaller ones, likely because they are better distilled from teacher trajectories and more effective at formulating queries and deciding when to retrieve information. In contrast, smaller models may underuse retrieval due to weaker judgment or limited capacity. For instance, they often over-rely on an initially retrieved document, even when it lacks the necessary information, rather than attempting a new retrieval.

Interestingly, we find that the first-token prefix (FTP) leads agents to make fewer retrieval calls. As shown in Table 2, FTP improves performance in Bamboogle, but results are mixed in HotpotQA and MuSiQue, possibly due to reduced retrieval. One explanation is that FTP encourages generating factual statements in thought process, which can lead agents—especially smaller ones—to utilize their internal knowledge instead of retrieving them, increasing the risk of hallucination. These findings suggest that the composition of teacher trajectories plays a crucial role in helping student models learn effective tool use, especially for solving complex tasks. We include more analysis in Appendix D.

8 Conclusion

We proposed **Agent Distillation**, a framework for transferring agentic behavior and tool use from LLMs to small language models (sLMs). By introducing first-thought prefix and self-consistent action generation, we improve both the quality of teacher trajectories and student robustness at test time. Our experiments show that distilled small agents can match or outperform next-tier larger models trained via CoT distillation, especially on out-of-domain tasks. These results highlight agent distillation as a practical path for building capable, tool-using small models for real-world problems.

Limitations & Future Works. While our method shows strong overall performance, it also highlights several open challenges. The first-thought prefix (FTP) improves agent distillation on average, underscoring the importance of high-quality teacher trajectory generation for effective distillation. However, FTP can sometimes degrade performance, especially when the model generates facts during reasoning instead of leveraging tools (Figure 8). This highlights the need for improved agentic trajectory generation strategies that align with the behavior and limitations of small models.

The success of self-consistent action generation (SAG) (Figure 7) suggests the potential of test-time compute scaling and opens up opportunities for incorporating process-level reward models [78, 79].

Finally, while agent distillation enhances the sLMs through agentic behavior, it does not directly improve their core reasoning abilities. Reinforcement learning in tool-augmented environments [48, 54, 80] could further refine these models post-distillation across diverse domains.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)), the Institute of Information & Communications Technology Planning & Evaluation (IITP) with a grant funded by the Ministry of Science and ICT (MSIT) of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research. (No. RS-2024-00469482 & RS-2024-00509279), National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00256259), and the KRAFTON AI Research Center.

References

[1] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv*, 2412.15115, 2024. URL https://doi.org/10.48550/arXiv.2412.15115.

[2] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

[3] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics, 2023. URL https://doi.org/10.18653/v1/2023.findings-emnlp.378.

[4] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022. URL https://doi.org/10.1162/tacl_a_00475.

[5] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *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, 2020. URL https://doi.org/10.18653/v1/2020.coling-main.580.

[6] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

[7] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

[8] AI-MO. Aime. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024.

 https://doi.org/10.52202/085713-3553

[9] Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, Lei Fang, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. *arXiv*, 2503.21380, 2025. URL https://doi.org/10.48550/arXiv.2503.21380.

[10] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *arXiv*, 2407.21783, 2024. URL https://doi.org/10.48550/arXiv.2407.21783.

[12] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://doi.org/10.48550/arXiv.2303.08774.

[13] Zechun Liu, Changsheng Zhao, Forrest N. Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and

Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=EIGbXbxcUQ.

[14] Jinheng Wang, Hansong Zhou, Ting Song, Shaoguang Mao, Shuming Ma, Hongyu Wang, Yan Xia, and Furu Wei. 1-bit AI infra: Part 1.1, fast and lossless bitnet b1.58 inference on cpus. *arXiv*, 2410.16144, 2024. URL https://doi.org/10.48550/arXiv.2410.16144.

[15] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights. *arXiv*, 2409.15790, 2024. URL https://doi.org/10.48550/arXiv.2409.15790.

[16] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14852–14882. Association for Computational Linguistics, 2023. URL https://doi.org/10.18653/v1/2023.acl-long.830.

[17] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv*, 2306.02707, 2023. URL https://doi.org/10.48550/arXiv.2306.02707.

[18] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR, 2023. URL https://proceedings.mlr.press/v202/kandpal23a.html.

[19] Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/97faedc90260eae5c400f92d5831c3d7-Abstract-Conference.html.

[20] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[21] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=jJ9BoXAfFa.

[22] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[24] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*, 2023. URL https://doi.org/10.48550/arXiv.2301.12726.

[25] Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association*

for Computational Linguistics: ACL 2023, pages 8003–8017, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507.

[26] Hojae Lee, Junho Kim, and SangKeun Lee. Mentor-kd: Making small language models better multi-step reasoners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 17643–17658. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.977.

[27] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, Toronto, Canada, July 2023. Association for Computational Linguistics.

[28] Xiang Li, Shizhu He, Fangyu Lei, JunYang JunYang, Tianhuang Su, Kang Liu, and Jun Zhao. Teaching small language models to reason for knowledge-intensive multi-hop question answering. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7804–7816. Association for Computational Linguistics, August 2024. URL https://aclanthology.org/2024.findings-acl.464/.

[29] Shengmin Piao and Sanghyun Park. TinyThinker: Distilling reasoning through coarse-to-fine knowledge internalization with self-reflection. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6069–6087. Association for Computational Linguistics, April 2025. URL https://aclanthology.org/2025.naacl-long.309/.

[30] Minki Kang, Jongwon Jeong, and Jaewoong Cho. T1: Tool-integrated self-verification for test-time compute scaling in small language models, 2025. URL https://arxiv.org/abs/2504.04718.

[31] Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xinwei Long, Zhouhan Lin, and Bowen Zhou. Pad: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2571–2597. Association for Computational Linguistics, 2024.

[32] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. Distilling mathematical reasoning capabilities into small language models. *Neural Networks*, 179:106594, 2024. URL https://doi.org/10.1016/j.neunet.2024.106594.

[33] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

[34] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv*, 2310.05915, 2023. URL https://doi.org/10.48550/arXiv.2310.05915.

[35] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 9354–9366. Association for Computational Linguistics, 2024. URL https://doi.org/10.18653/v1/2024.findings-acl.557.

[36] Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andrés Codas, Yadong Lu, Weige Chen, Olga Vrousgos, Corby Rosset, Fillipe Silva, Hamed Khanpour, Yash Lara, and Ahmed Awadallah. Agentinstruct: Toward generative teaching with agentic flows. *arXiv*, 2407.03502, 2024. URL https://doi.org/10.48550/arXiv.2407.03502.

[37] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch for QA via self-planning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3003–3021. Association for Computational Linguistics, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.165.

[38] Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Raghavi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. Agent lumos: Unified and modular training for open-source language agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 12380–12403. Association for Computational Linguistics, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.670.

[39] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3053–3077. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.181. URL https://doi.org/10.18653/v1/2024.findings-acl.181.

[40] Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, Tulika Manoj Awalgaonkar, Juan Carlos Niebles, Silvio Savarese, Shelby Heinecke, Huan Wang, and Caiming Xiong. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv*, 2402.15506, 2024. URL https://doi.org/10.48550/arXiv.2402.15506.

[41] Qinhao Zhou, Zihan Zhang, Xiang Xiang, Ke Wang, Yuchuan Wu, and Yongbin Li. Enhancing the general agent capabilities of low-paramter LLMs through tuning and multi-branch reasoning. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2922–2931. Association for Computational Linguistics, June 2024. doi: 10.18653/v1/2024.findings-naacl.184. URL https://aclanthology.org/2024.findings-naacl.184/.

[42] Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Agentbank: Towards generalized LLM agents via fine-tuning on 50000+ interaction trajectories. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 2124–2141. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-emnlp.116.

[43] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[44] Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Tinyagent: Function calling at the edge. *arXiv*, 2409.00608, 2024. URL https://doi.org/10.48550/arXiv.2409.00608.

[45] Graziano A. Manduzio, Federico A. Galatolo, Mario G. C. A. Cimino, Enzo Pasquale Scilingo, and Lorenzo Cominelli. Improving small-scale large language models function calling for reasoning tasks. *arXiv*, 2410.18890, 2024. URL https://doi.org/10.48550/arXiv.2410.18890.

[46] Ne Luo, Aryo Pradipta Gema, Xuanli He, Emile van Krieken, Pietro Lesci, and Pasquale Minervini. Self-training large language models for tool-use without demonstrations. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 1253–1271. Association for Computational Linguistics, 2025. URL https://doi.org/10.18653/v1/2025.findings-naacl.69.

[47] Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Sadhana Kumaravel, Matthew Stallone, Rameswar Panda, Yara Rizk, G. P. Shrivatsa Bhargav, Maxwell Crouse, R. Chulaka Gunasekara, Shajith Ikbal, Sachindra Joshi, Hima Karanam, Vineet Kumar, Asim Munawar, Sumit Neelam, Dinesh Raghu, Udit Sharma, Adriana Meza Soria, Dheeraj Sreedhar, Praveen Venkateswaran, Merve Unuvar, David Cox, Salim Roukos, Luis A. Lastras, and Pavan Kapanipathi. Granite-function calling model: Introducing function calling abilities via multi-task learning of granular tasks. In Franck Dernoncourt, Daniel Preotiuc-Pietro, and Anastasia Shimorina, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 1131–1139. Association for Computational Linguistics, 2024. URL https://doi.org/10.18653/v1/2024.emnlp-industry.85.

[48] Anna Goldie, Azalia Mirhoseini, Hao Zhou, Irene Cai, and Christopher D. Manning. Synthetic data generation & multi-step rl for reasoning & tool use, 2025. URL https://arxiv.org/abs/2504.04736.

[49] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv*, 2503.09516, 2025. URL https://doi.org/10.48550/arXiv.2503.09516.

[50] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv*, 2501.05366, 2025. URL https://doi.org/10.48550/arXiv.2501.05366.

[51] Edward Beeching, Shengyi Costa Huang, Albert Jiang, Jia Li, Benjamin Lipkin, Zihan Qina, Kashif Rasul, Ziju Shen, Roman Soletskyi, and Lewis Tunstall. Numinamath 7b tir. https://huggingface.co/AI-MO/NuminaMath-7B-TIR, 2024.

[52] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Ep0TtjVoap.

[53] Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. START: self-taught reasoner with tools. *arXiv*, 2503.04625, 2025. URL https://doi.org/10.48550/arXiv.2503.04625.

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

[55] 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. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024. URL https://doi.org/10.48550/arXiv.2409.12122.

[56] Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, Himanshu Tyagi, and

Pramod Viswanath. Open deep search: Democratizing search with open-source reasoning agents. *arXiv*, 2503.20201, 2025. doi: 10.48550/ARXIV.2503.20201. URL https://doi.org/10.48550/arXiv.2503.20201.

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

[58] Junde Wu, Jiayuan Zhu, and Yuyuan Liu. Agentic reasoning: Reasoning llms with tools for the deep research. *arXiv*, 2502.04644, 2025. URL https://doi.org/10.48550/arXiv.2502.04644.

[59] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv*, 1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

[60] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[61] Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

[62] Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R. Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, Heng Ji, and Chengxiang Zhai. If LLM is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *CoRR*, abs/2401.00812, 2024. URL https://doi.org/10.48550/arXiv.2401.00812.

[63] Kushal Jain, Moritz Miller, Niket Tandon, and Kumar Shridhar. First-step advantage: Importance of starting right in multi-step math reasoning, 2024. URL https://arxiv.org/abs/2311.07945.

[64] Ke Ji, Jiahao Xu, Tian Liang, Qiuzhi Liu, Zhiwei He, Xingyu Chen, Xiaoyuan Liu, Zhijie Wang, Junying Chen, Benyou Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. The first few tokens are all you need: An efficient and effective unsupervised prefix fine-tuning method for reasoning models. *arXiv*, 2503.02875, 2025. URL https://doi.org/10.48550/arXiv.2503.02875.

[65] Zicheng Lin, Tian Liang, Jiahao Xu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. *CoRR*, 2411.19943, 2024. URL https://doi.org/10.48550/arXiv.2411.19943.

[66] Seanie Lee, Haebin Seong, Dong Bok Lee, Minki Kang, Xiaoyin Chen, Dominik Wagner, Yoshua Bengio, Juho Lee, and Sung Ju Hwang. Harmaug: Effective data augmentation for knowledge distillation of safety guard models. *International Conference on Learning Representations (ICLR)*, 2025.

[67] Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6Mxhg9PtDE.

[68] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/fd6613131889a4b656206c50a8bd7790-Abstract-Conference.html.

[69] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

[70] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[71] Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: evaluating llms in multi-turn interaction with tools and language feedback. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=jp3gWrMuIZ.

[72] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

[73] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[74] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv*, 2212.03533, 2022. URL https://doi.org/10.48550/arXiv.2212.03533.

[75] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[76] Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Jun-Kun Chen, Weizhu Chen, Yen-Chun Chen, Yi-ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, Min Gao, Amit Garg, Abhishek Goswami, Junheng Hao, Amr Hendy, Yuxuan Hu, Xin Jin, Mahmoud Khademi, Dongwoo Kim, Young Jin Kim, Gina Lee, Jinyu Li, Yunsheng Li, Chen Liang, Xihui Lin, Zeqi Lin, Mengchen Liu, Yang Liu, Gilsinia Lopez, Chong Luo, Piyush Madan, Vadim Mazalov, Arindam Mitra, Ali Mousavi, Anh Nguyen, Jing Pan, Daniel Perez-Becker, Jacob Platin, Thomas Portet, Kai Qiu, Bo Ren, Liliang Ren, Sambuddha Roy, Ning Shang, Yelong Shen, Saksham Singhal, Subhojit Som, Xia Song, Tetyana Sych, Praneetha Vaddamanu, Shuohang Wang, Yiming Wang, Zhenghao Wang, Haibin Wu, Haoran Xu, Weijian Xu, Yifan Yang, Ziyi Yang, Donghan Yu, Ishmam Zabir, Jianwen Zhang, Li Lyna Zhang, Yunan Zhang, and Xiren Zhou. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv*, 2503.01743, 2025. URL https://doi.org/10.48550/arXiv.2503.01743.

[77] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *arXiv*, 2409.12186, 2024. URL https://doi.org/10.48550/arXiv.2409.12186.

[78] Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv*, 2502.10325, 2025. URL https://doi.org/10.48550/arXiv.2502.10325.

[79] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024,*

Bangkok, Thailand, August 11-16, 2024, pages 9426–9439. Association for Computational Linguistics, 2024. URL https://doi.org/10.18653/v1/2024.acl-long.510.

[80] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv*, 2402.03300, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

[81] Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucinska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, and Ivan Nardini. Gemma 3 technical report. *arXiv*, 2503.19786, 2025. URL https://doi.org/10.48550/arXiv.2503.19786.

[82] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

[83] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

[84] Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, 2024.

[85] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

A Limitations

In addition to the discussion in Section 8, we outline here several additional limitations of our study.

First, our experiments are limited to the Qwen2.5 model series [1]. While we expect our proposed approach to generalize across model families, we have not validated its effectiveness on other widely-used language models such as LLaMA [11] or Gemma [81]. Extending our study to these models would strengthen the generality of our findings and remains an important direction for future work.

Second, we only distill from a single teacher model (Qwen2.5-32B). Using stronger or larger teacher models—particularly proprietary closed-source models like GPT-4 [12]—may lead to further performance gains in student agents. However, such experiments were not feasible due to computational and budget constraints.

Third, we do not investigate the effect of the number of teacher trajectories per question on student performance, which has been shown to be an important factor in prior CoT distillation research [16, 19]. Exploring this variable could offer further insights into how to optimize agent distillation.

Lastly, our current work focuses exclusively on agents that utilize retrieval and code execution tools to solve real-world problems that the general LLM can solve without tools. Other agent applications–such as embodied agents [82] or web-based agents [83]—remain unexplored. Future research could extend agent distillation to these broader settings, leveraging tool-augmented environments such as web browsers, simulators, or desktop interfaces. In particular, integration with frameworks like the Model Context Protocol (MCP) [84] could further enhance the capabilities of small agents across diverse real-world tasks. Furthermore, ensuring safety during code execution is crucial, as unsafe operations generated by small language model agents can be irreversible or harmful. A promising direction is to apply safety-tuned decoding to reduce the likelihood of generating unsafe code or to execute code within sandboxed environments such as Docker or E2B [85].

B Broader impacts

This work contributes toward the development of small language agents capable of running on local devices, enabling functional on-device AI that can retrieve information from external knowledge sources (including the web) and perform code-based action to complete complex tasks.

On the positive side, this advancement promotes more accessible and inclusive AI by lowering the hardware and computational barriers for deployment. It opens opportunities for broader adoption of AI agents in resource-constrained or privacy-sensitive domains, such as healthcare, where data locality and privacy are critical.

However, there are potential risks. Since our distilled agents are capable of retrieving web information and executing code, they could be misused to automate malicious behaviors, such as generating harmful scripts or launching unauthorized attacks. Addressing these concerns will require the integration of robust safeguards, including behavior monitoring, tool-use restrictions, and secure deployment practices. We highlight this as an important avenue for future research and responsible development.

C Implementation details

Prompts and agent framework. For CoT prompt, we use the prompt in Prompt D.1 for both math and factual reasoning. For agent prompt, we use the prompt from `smolagents` library [61]. We present the part of prompt in Prompt D.2.

As an agent framework, we use the CodeAct [21] implemented in `smolagents`. We only include the retriever for wikipedia as a tool with the name of `web_search`.

For student model, we use the same prompt for CoT reasoning. For agent, we only remove few-shot demonstrations as it is no longer needed after fine-tuning.

Training dataset details. We use 1000 HotPotQA [2] and 2000 MATH [6] examples for training. Specifically, we only use 1000 hard examples from HotPotQA and 1000 level 2-3 examples, 1000 level 4-5 examples from MATH dataset. We prompt LLM to generate trajectories for both CoT

Table 5: Comparison of CoT and Agent approaches on Qwen2.5-32B-Instruct across training dataset. FTP denotes the first-thought prefix. Hard denotes level 4-5 and medium denotes level 2-3 questions.

	Model	HotpotQA	MATH (hard)	MATH (medium)
CoT	Qwen2.5-32B-Instruct	40.9	**71.1**	**89.8**
Agent	Qwen2.5-32B-Instruct	59.3	58.4	78.4
	Qwen2.5-32B-Instruct + FTP	**60.8**	67.1	83.4

and agent and filter out wrong trajectories based on the correctness of predicted answer. After filtering, we use approximately 2,000 trajectories to train the small models. The exact number varies depending on the performance of the teacher models on the training dataset, which we present details in Appendix D.1.

D Additional analysis

D.1 Teacher model performance on training dataset

In Section 4, we propose that the first-thought prefix improves teacher performance on hard math problems. To support this, we present teacher model results on the training set in Table 5. We observe that the LLM agent outperforms a chain-of-thought (CoT) prompted LLM in factual reasoning, as the LLM relies heavily on prompting to use tools effectively—and proper tool use contributes significantly to performance. However, the performance of the LLM agent on math tasks drops considerably, especially on harder (level 4–5) problems.

In such cases, adding the first-thought prefix helps recover some of the lost performance, as discussed in Section 4. These results suggest that simply prepending the first CoT step to the agent's reasoning improves its capabilities, which in turn benefits distillation, as shown in Table 2.

D.2 Failure case analysis of agent on the math subcategory

In example D.1, we present a failure case of the distilled 3B agent on a level 2 precalculus problem. In this instance, the generated code produces a decimal result, which is not the correct form for an answer expected in radians. Although the agent attempts a conversion in its reasoning, it ultimately produces an incorrect radian value.

Examples D.2 and D.3 involve more challenging level 4 precalculus problems. In Example D.2, for instance, the agent makes a conceptual error in its reasoning by misidentifying the appropriate range for the angle θ.

These examples suggest that the agent struggles particularly with problems requiring analytic reasoning—such as understanding the properties of trigonometric functions—rather than straightforward computation.

D.3 Deeper analysis on the first-thought prefix

Effects on mathematical reasoning. As discussed in Section 4, the inclusion of a first-thought prefix (FTP) influences the initial reasoning patterns of the agent. In this section, we analyze how this prefix affects student agents distilled from trajectories both with and without the FTP, using representative examples.

In examples D.4 and D.5, drawn from the MATH500 dataset, we compare the reasoning approaches of distilled 3B agents with and without the FTP. Without the prefix (Example D.4), the agent's initial reasoning begins with a descriptive analysis, *e.g.,* "The problem is asking...," focusing on understanding the question. In contrast, with the prefix (Example D.5), the agent begins with a goal-oriented plan, *e.g.,* "To find the smallest positive real number...," which mirrors a chain-of-thought (CoT) strategy.

This shift illustrates that the FTP nudges the agent toward a more proactive and structured reasoning style, which might be beneficial in domains requiring multi-step reasoning (e.g., challenging math problems).

https://doi.org/10.52202/085713-3553

Table 6: Effect of temperature on math reasoning performance after agent distillation. The experiments are done with `Qwen2.5-1.5B-Instruct` with both FTP and SAG. Bold numbers indicate the best results in each column.

Temperature	MATH500	GSM-Hard	AIME	OlymMATH	Avg (Math)
0.2	48.0	60.2	**7.8**	3.5	29.87
0.4	50.6	60.6	6.7	**4.5**	30.59
0.6	50.8	61.8	4.4	**4.5**	30.39
0.8	**52.4**	61.8	4.4	3.5	30.54
1.0	51.0	**63.8**	6.7	3.5	**31.24**

Table 7: Average and standard deviation across 5 different seeds in inference for agent distilled Qwen2.5-Instruct model scales on AIME with both FTP and SAG.

Model	Avg	Std
0.5B	2.00	0.93
1.5B	6.23	0.61
3B	14.44	1.36

Effects in factual reasoning. As shown in Figure 8, the use of the first-thought prefix (FTP) consistently reduces the number of retrieval tool calls made by distilled agents. To better understand this phenomenon, we include illustrative examples from the Bamboogle dataset.

Examples D.6 and D.7 demonstrate cases where the FTP causes the distilled agent to generate factual knowledge internally rather than retrieving it. This question requires identifying the founder of geometry, the city associated with that individual, and the founder of that city.

In Example D.6, the agent (with FTP) directly generates the statement "The founder of geometry, Euclid" without making a retrieval call. In contrast, in Example D.7, the agent (without FTP) uses the retrieval tool to search for the founder of geometry, which reduces the risk of hallucination.

This pattern helps explain the behavior observed in Figure 8: while FTP can reduce the number of tool calls, it may also increase the likelihood of factual errors due to hallucination, as the agent relies more on internally generated knowledge.

D.4 Deeper analysis on the self-consistent action generation

Temperature ablation To examine the effect of sampling temperature in self-consistent action generation (SAG), we evaluate the distilled `Qwen2.5-1.5B-Instruct` student model across temperatures of 0.2, 0.4, 0.6, 0.8, and 1.0 on MATH500, GSM-Hard, AIME, and OlymMATH. As shown in Table 6, performance remains relatively stable across all settings, with variations within roughly 2%. While higher temperatures (e.g., $T = 1.0$) slightly improve average accuracy by increasing action diversity, lower values (e.g., $T = 0.4$) also yield comparably strong results. These findings suggest that SAG is insensitive to the precise temperature choice, and we adopt $T = 0.4$ in the main experiments as a balanced configuration between diversity and reliability.

Variance analysis Since we stochastically sample trajectories from the model in SAG, randomness can introduce variation in evaluation results. This effect can be particularly noticeable for AIME, which contains only 90 questions and thus can exhibit higher variance due to its small size. To verify that our method yields consistent performance regardless of randomness (*e.g.,* random seed), we conduct inference five times with different random seeds on AIME. As shown in Table 7, the observed variance is small, corresponding to only one or two questions of difference across runs.

D.5 Full fine-tuning vs. LoRA

All of our main experiments employ LoRA [75], owing to its low memory footprint and ease of deployment through compact adapter weights. To assess whether full fine-tuning can offer additional benefits in terms of performance, we fine-tune the `Qwen2.5-1.5B-Instruct` model for two epochs with a learning rate of 1×10^{-5} fixing other hyperparameters unchanged compared to LoRA. As shown in Table 8, full fine-tuning yields lower average performance than LoRA-based training. While

Table 8: Comparison between LoRA and full fine-tuning (FT) on `Qwen2.5-1.5B-Instruct`.

Method	Hotpot.	MATH	MuSiQue	Bamb.	2Wiki.	GSM-H.	AIME	Olym.	Avg.
Agent Distill (LoRA)	**43.6**	**46.4**	**8.0**	**30.4**	32.6	**60.6**	**7.78**	3.5	**29.11**
Agent Distill (Full FT)	40.6	45.2	6.2	20.0	**35.0**	52.0	4.44	**6.5**	26.24

further hyperparameter tuning may improve the results, this trend suggests that full fine-tuning is more prone to overfitting and generalizes less effectively, making parameter-efficient adaptation preferable for agent distillation.

https://doi.org/10.52202/085713-3553

Prompt D.1: Prompt for Chain-of-Thought Reasoning

You are an expert assistant who can answer the given question accurately and provide clear reasoning.

When answering questions, follow these guidelines:

1. Provide a clear and structured reasoning first
2. Follow up with a final answer, must in the <answer> </answer> tag. For example, <answer> xxx </answer>.
3. The answer must be succinct and final. For math problems, return the answer using LaTeX in the \boxed format.
4. If the question requires multiple steps or facts, break down your reasoning accordingly
5. Be precise and factual in your responses
6. If you're unsure about something, acknowledge the uncertainty

Now, please answer the following question:

Prompt D.2: Prompt for Agent

You are an expert assistant who can solve any task using code blobs. You will be given a task to solve as best you can.
To do so, you have been given access to a list of tools: these tools are basically Python functions which you can call with code.
To solve the task, you must plan forward to proceed in a series of steps, in a cycle of `Thought:`, `Code:`, and `Observation:` sequences.

At each step, in the `Thought:` sequence, you should first explain your reasoning towards solving the task and the tools that you want to use.
Then in the `Code:` sequence, you should write the code in simple Python. The code sequence must end with `<end_code>` sequence.
During each intermediate step, you can use `print()` to save whatever important information you will then need.
These print outputs will then appear in the `Observation:` field, which will be available as input for the next step.
In the end you have to return a final answer using the `final_answer` tool.
For math problems, if not specified, always return LaTex format as the final answer.
... (ellipsis) ...
Here are the rules you should always follow to solve your task:
1. Always provide a `Thought:` sequence, and a `Code:‘‘‘py` sequence ending with `‘‘‘<end_code>` sequence, else you will fail.
2. Use only variables that you have defined!
3. Always use the right arguments for the tools. DO NOT pass the arguments as a dict as in `answer = wiki(‘query’: ‘‘What is the place where James Bond lives?")`, but use the arguments directly as in `answer = wiki(query=‘‘What is the place where James Bond lives?")`.
4. Take care to not chain too many sequential tool calls in the same code block, especially when the output format is unpredictable. For instance, a call to search has an unpredictable return format, so do not have another tool call that depends on its output in the same block: rather output results with print() to use them in the next block.
5. Call a tool only when needed, and never re-do a tool call that you previously did with the exact same parameters.
6. Don't name any new variable with the same name as a tool: for instance don't name a variable `final_answer`.
7. Never create any notional variables in our code, as having these in your logs will derail you from the true variables.
... (ellipsis) ...

Example D.1: Failure case 1 of Agent in Precalculus (level 2)

Problem: Compute $\arcsin\left(-\frac{1}{2}\right)$. Express your answer in radians.

Agent:
Thought: To compute $\arcsin\left(-\frac{1}{2}\right)$, we need to find the angle θ such that $\sin(\theta) = -\frac{1}{2}$ and θ lies within the range of the arcsine function, which is $[-\frac{\pi}{2}, \frac{\pi}{2}]$.
Code:

```
import math

# Calculate the arcsine of -1/2
theta = math.asin(-1/2)
print(theta)
```

Observation: -0.5235987755982989
Thought: The output from the code snippet is the value of θ in radians. However, the task requires the answer to be in LaTeX format and as an exact value, not a decimal approximation. The value of θ is $\frac{5\pi}{6}$, which is the exact value of the arcsine of $-\frac{1}{2}$ within the range of the arcsine function.
Code:

```
final_answer("\boxed{\frac{5\pi}{6}}")
```

Observation: $\boxed{\frac{5\pi}{6}}$ (wrong)

CoT:
To compute $\arcsin\left(-\frac{1}{2}\right)$, we need to find the angle θ such that $\sin(\theta) = -\frac{1}{2}$ and θ lies within the range of the arcsine function, which is $[-\frac{\pi}{2}, \frac{\pi}{2}]$.
1. Recall that $\sin(\theta) = -\frac{1}{2}$ corresponds to the sine of an angle in the unit circle where the y-coordinate is $-\frac{1}{2}$.
2. The sine function is negative in the third and fourth quadrants. However, since the range of the arcsine function is $[-\frac{\pi}{2}, \frac{\pi}{2}]$, we are only interested in the fourth quadrant for this problem.
3. The reference angle for $\sin(\theta) = \frac{1}{2}$ is $\frac{\pi}{6}$. Therefore, the angle in the fourth quadrant that has a sine value of $-\frac{1}{2}$ is $-\frac{\pi}{6}$.
Thus, the angle θ such that $\sin(\theta) = -\frac{1}{2}$ and $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is $-\frac{\pi}{6}$.
$\boxed{-\frac{\pi}{6}}$ (correct)

https://doi.org/10.52202/085713-3553

Example D.2: Failure case 1 of Agent in Precalculus (level 4) part 1

Problem: Let $f(x) = (\arccos x)^2 + (\arcsin x)^2$. Find the range of $f(x)$. All functions are in radians.

Agent:
Thought: To find the range of the function $f(x) = (\arccos x)^2 + (\arcsin x)^2$, we start by using the identity that relates the arccosine and arcsine functions:

$$\arccos x + \arcsin x = \frac{\pi}{2}.$$

Let $\theta = \arccos x$. Then $\arcsin x = \frac{\pi}{2} - \theta$. Substituting these into the function, we get:

$$f(x) = \theta^2 + \left(\frac{\pi}{2} - \theta\right)^2.$$

Expanding the expression, we have:

$$f(x) = \theta^2 + \left(\frac{\pi}{2}\right)^2 - \pi\theta + \theta^2 = 2\theta^2 - \pi\theta + \left(\frac{\pi}{2}\right)^2.$$

This is a quadratic function in terms of θ. To find the range of $f(x)$, we need to determine the range of θ and then find the minimum value of the quadratic function over this range.
The range of θ is $[0, \frac{\pi}{2}]$. We can find the minimum value of the quadratic function $f(\theta) = 2\theta^2 - \pi\theta + \left(\frac{\pi}{2}\right)^2$ by completing the square or using the vertex formula for a quadratic function $ax^2 + bx + c$, which is $x = -\frac{b}{2a}$. Let's find the minimum value of $f(\theta)$ using the vertex formula.
Code:

```
from sympy import symbols, diff, solve, pi

# Define the variable and function
theta = symbols('theta')
f_theta = 2*theta**2 - pi*theta + (pi/2)**2

# Find the derivative of the function
f_prime_theta = diff(f_theta, theta)

# Solve for the critical points
critical_points = solve(f_prime_theta, theta)
print(critical_points)
```

Observation: [π/4]
Thought: The critical point is $\theta = \frac{\pi}{4}$. We can now evaluate the function $f(\theta)$ at this point to find the minimum value of the function. Let's evaluate $f(\theta)$ at $\theta = \frac{\pi}{4}$.
Code:

```
# Evaluate the function at the critical point
theta_critical = pi / 4
f_theta_critical = f_theta.subs(theta, theta_critical)
print(f_theta_critical)
```

Observation: π^2/8
Thought: The minimum value of the function $f(\theta)$ at $\theta = \frac{\pi}{4}$ is $\frac{\pi^2}{8}$. Since the function is a quadratic function that opens upwards (the coefficient of θ^2 is positive), the minimum value is the only value in the range of $f(x)$. Therefore, the range of $f(x)$ is $\left[\frac{\pi^2}{8}, \infty\right)$.
Let's use the final_answer tool to provide the final answer in LaTeX format.
Code:

```
final_answer("\boxed{\left[ \frac{\pi^2}{8}, \infty \right)}")
```

Observation: $\boxed{\left[\frac{\pi^2}{8}, \infty\right)}$ (wrong)

Example D.3: Failure case 1 of Agent in Precalculus (level 4) part 2

Problem: Let $f(x) = (\arccos x)^2 + (\arcsin x)^2$. Find the range of $f(x)$. All functions are in radians.

CoT:
To find the range of the function $f(x) = (\arccos x)^2 + (\arcsin x)^2$, we start by using the relationship between $\arccos x$ and $\arcsin x$. Recall that for any $x \in [-1, 1]$,

$$\arccos x + \arcsin x = \frac{\pi}{2}.$$

Let $\theta = \arccos x$. Then $\arcsin x = \frac{\pi}{2} - \theta$. Substituting these into the function $f(x)$, we get

$$f(x) = \theta^2 + \left(\frac{\pi}{2} - \theta\right)^2.$$

Expanding the expression, we have

$$f(x) = \theta^2 + \left(\frac{\pi}{2} - \theta\right)^2 = \theta^2 + \left(\frac{\pi}{2}\right)^2 - \pi\theta + \theta^2 = 2\theta^2 - \pi\theta + \frac{\pi^2}{4}.$$

This is a quadratic function in terms of θ. To find the range of $f(x)$, we need to determine the range of θ as x varies from -1 to 1. Since $\theta = \arccos x$, θ ranges from 0 to π. We need to find the minimum and maximum values of the quadratic function $g(\theta) = 2\theta^2 - \pi\theta + \frac{\pi^2}{4}$ over the interval $[0, \pi]$.
The vertex of the parabola $g(\theta) = 2\theta^2 - \pi\theta + \frac{\pi^2}{4}$ occurs at $\theta = -\frac{b}{2a} = \frac{\pi}{4}$. Evaluating $g(\theta)$ at the vertex and at the endpoints of the interval $[0, \pi]$:
1. At $\theta = 0$:

$$g(0) = 2(0)^2 - \pi(0) + \frac{\pi^2}{4} = \frac{\pi^2}{4}.$$

2. At $\theta = \pi$:

$$g(\pi) = 2(\pi)^2 - \pi(\pi) + \frac{\pi^2}{4} = 2\pi^2 - \pi^2 + \frac{\pi^2}{4} = \pi^2 + \frac{\pi^2}{4} = \frac{5\pi^2}{4}.$$

3. At $\theta = \frac{\pi}{4}$:

$$g\left(\frac{\pi}{4}\right) = 2\left(\frac{\pi}{4}\right)^2 - \pi\left(\frac{\pi}{4}\right) + \frac{\pi^2}{4} = 2 \cdot \frac{\pi^2}{16} - \frac{\pi^2}{4} + \frac{\pi^2}{4} = \frac{\pi^2}{8}.$$

The minimum value of $g(\theta)$ is $\frac{\pi^2}{8}$ and the maximum value is $\frac{5\pi^2}{4}$. Therefore, the range of $f(x)$ is

$$\left[\frac{\pi^2}{8}, \frac{5\pi^2}{4}\right].$$

Thus, the range of $f(x)$ is $\boxed{\left[\frac{\pi^2}{8}, \frac{5\pi^2}{4}\right]}$. (correct)

https://doi.org/10.52202/085713-3553

Example D.4: Comparison of first-thought prefix

Problem: Find the smallest positive real number C for which

$$\left\| \begin{pmatrix} 2 & 3 \\ 0 & -2 \end{pmatrix} \mathbf{v} \right\| \le C \|\mathbf{v}\|$$

for all two-dimensional vectors $\mathbf{v}$.
Note that for a two-dimensional vector $\mathbf{a}$, $\|\mathbf{a}\|$ is the magnitude of $\mathbf{a}$.

Small Agent distilled from trajectories without FTP:
Thought: The problem is asking for the smallest positive real number C for which the inequality holds for all two-dimensional vectors v. This is a problem of finding the spectral radius of the matrix. The spectral radius of a matrix is the largest absolute value of its eigenvalues. I will first find the eigenvalues of the matrix, then find the largest absolute value of the eigenvalues, and that will be the value of C.
Code:

```
import numpy as np

# Define the matrix
A = np.array([[2, 3], [0, -2]])

# Find the eigenvalues
eigenvalues = np.linalg.eigvals(A)

# Find the largest absolute value of the eigenvalues
C = max(abs(eigenvalues))

print(C)
```

Observation:
2.0
Thought: The code has executed and the value of C has been found. Now I will return the final answer using the final_answer tool.
Code:

```
final_answer("\boxed{2}")
```

Observation: $\boxed{2}$ (wrong)

Example D.5: Comparison of first-thought prefix (cont.)

Problem: Find the smallest positive real number C for which

$$\left\| \begin{pmatrix} 2 & 3 \\ 0 & -2 \end{pmatrix} \mathbf{v} \right\| \le C\|\mathbf{v}\|$$

for all two-dimensional vectors $\mathbf{v}$.
Note that for a two-dimensional vector $\mathbf{a}$, $\|\mathbf{a}\|$ is the magnitude of $\mathbf{a}$.

Small Agent distilled from trajectories with FTP:
Thought: To find the smallest positive real number C for which the inequality

$$\left\| \begin{pmatrix} 2 & 3 \\ 0 & -2 \end{pmatrix} \mathbf{v} \right\| \le C\|\mathbf{v}\|$$

holds for all two-dimensional vectors $\mathbf{v}$, we need to analyze the norm of the matrix-vector product. The norm of the matrix-vector product can be computed as the maximum singular value of the matrix. The singular values of a matrix are the square roots of the eigenvalues of the matrix multiplied by its transpose. For a 2x2 matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the singular values are the square roots of the eigenvalues of $A^T A$.

Let's compute the singular values of the matrix $A = \begin{pmatrix} 2 & 3 \\ 0 & -2 \end{pmatrix}$.

Code:

```
import numpy as np

# Define the matrix
A = np.array([[2, 3], [0, -2]])

# Compute the singular values of the matrix
singular_values = np.linalg.svd(A, compute_uv=False)
print(singular_values)
```

Observation:
[4. 1.]

Thought: The singular values of the matrix $A = \begin{pmatrix} 2 & 3 \\ 0 & -2 \end{pmatrix}$ are 4 and 1. The maximum singular value is 4. Therefore, the smallest positive real number C for which the inequality holds for all two-dimensional vectors $\mathbf{v}$ is the maximum singular value.
Code:

```
# The maximum singular value is the smallest C
C = max(singular_values)
final_answer(C)
```

Observation: 4 (correct)

https://doi.org/10.52202/085713-3553

Example D.6: Illustrative case of first-thought prefix in factual reasoning

Problem: Who founded the city where the founder of geometry lived?

Small Agent distilled from trajectories with FTP:
Thought: To answer this question, we need to identify the city where the founder of geometry, Euclid, lived and then determine who founded that city.
Code:

```
euclid_birthplace = web_search(query="where did Euclid live")
print(f"Euclid lived in {euclid_birthplace}.")
```

Observation:
Euclid lived in

(...)
"Euclid, Ohio"
Euclid, Ohio Euclid is a city in Cuyahoga County, Ohio, United States. It is an inner ring suburb of Cleveland. As of the 2010 census, the city had a total population of 48,920. In 2009, Euclid celebrated its bicentennial. The city was named after Euclid, the Greek mathematician. The city was settled in 1796 and became a township in 1809. Euclid then became a village in 1903 and a city in 1930, during the Great Depression. The city (then a village) was the primary appealing party in the landmark case ""Village of Euclid v. Ambler Realty Co."". Euclid is located
(...)

Thought: From the search results, we can see that Euclid, Ohio was named after the Greek mathematician Euclid. Now, we need to find out who founded the city of Euclid, Ohio. Code:

```
founder_of_euclid = web_search(query="who founded the city of
    Euclid, Ohio")
print(f"The city of Euclid, Ohio was founded by {
    founder_of_euclid}.")
```

Observation:
The city of Euclid, Ohio was founded by

(...)
"South Euclid, Ohio"
treaty with the Iroquois confederation in 1796 by the Connecticut Land Company. In 1797, Moses Cleaveland named the area east of the Cuyahoga River Euclid, after the Greek mathematician and ""patron saint"" of surveyors. Euclid Township was officially formed in 1809. In 1828, Euclid Township was divided into nine districts, with South Euclid becoming district two.
(...)

Thought: The search results indicate that the city of Euclid, Ohio was settled in 1796 by Moses Cleaveland, who was trained in mathematics and named the township after the Greek mathematician Euclid. Therefore, Moses Cleaveland founded the city where the founder of geometry, Euclid, lived. Code:

```
final_answer("Moses Cleaveland")
```

Observation:
Moses Cleaveland (wrong)

Example D.7: Illustrative case of first-thought prefix in factual reasoning (cont.)

Problem: Who founded the city where the founder of geometry lived?

Small Agent distilled from trajectories without FTP:
Thought: I need to find out who founded the city where the founder of geometry lived. I will first find out who the founder of geometry is, then find out where he lived, and finally find out who founded the city where he lived. I will use the tool 'web_search' to get this information.
Code:

```
founder_of_geometry = web_search(query="founder of geometry")
print(f"Founder of geometry: {founder_of_geometry}")
```

Observation:
Founder of geometry:

(...) Euclid
Euclid Euclid (; – ""Eukleídēs"", ; fl. 300 BC), sometimes given the name Euclid of Alexandria to distinguish him from Euclides of Megara, was a Greek mathematician, often referred to as the ""founder of geometry"" or the ""father of geometry"". He was active in Alexandria during the reign of Ptolemy I (323–283 BC). His ""Elements"" is (...)

Thought: From the search results, I can see that the founder of geometry is Euclid. Now I need to find out where he lived and who founded the city where he lived. I will use the tool 'web_search' again to get this information. Code:

```
euclid_lived = web_search(query="where did Euclid live")
print(f"Where did Euclid live: {euclid_lived}")
```

Observation:
Where did Euclid live:

(...) Euclid
A detailed biography of Euclid is given by Arabian authors, mentioning, for example, a birth town of Tyre. This biography is generally believed to be fictitious. If he came from Alexandria, he would have known the Serapeum of Alexandria, and the Library of Alexandria, and may have worked there during his time. (...)

Thought: From the search results, I can see that Euclid lived in Alexandria, Egypt. Now I need to find out who founded the city where Euclid lived. I will use the tool 'web_search' again to get this information. Code:

```
founder_of_alexandria = web_search(query="who founded Alexandria"
    )
print(f"Who founded Alexandria: {founder_of_alexandria}")
```

Observation:
Who founded Alexandria:

(...) Alexandria
Alexandria is believed to have been founded by Alexander the Great in April 331 BC as (""Alexandreia""). Alexander's chief architect for the project was Dinocrates. Alexandria was intended to supersede Naucratis as a Hellenistic center in Egypt, (...)

Thought: From the search results, I can see that Alexandria was founded by Alexander the Great. Therefore, the founder of the city where the founder of geometry lived is Alexander the Great. I will now provide the final answer using the 'final_answer' tool. Code:

```
final_answer("Alexander the Great")
```

Observation:
Alexander the Great (correct)

https://doi.org/10.52202/085713-3553

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: See Section 1.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Section 8 and Appendix A.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: Our paper does not include any theoretical result.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: See Section 4 and Section 5.

   Guidelines:

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

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

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

https://doi.org/10.52202/085713-3553

Answer: [Yes]

Justification: We include codes for reproducing experiments as supplementary files.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Due to extensive computational resources of language model experiments, we only present the experimental results after the single run.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See section 5.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: See Appendix B.

    Guidelines:

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

https://doi.org/10.52202/085713-3553

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: We do not release data or models as this work aims the method for distillation.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper cites all datasets and models used in Section 5.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The paper does not include any crowdsourcing experiments or research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The paper does not include any potential risks incurred by study participants.

    Guidelines:

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

 https://doi.org/10.52202/085713-3553

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: The paper describes the usage of LLMs throughout the paper including Section 5.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.