SmartAD: Capacity-Aligned Agent Distillation for Small Language Models

Guokai Tang and Feng Zhao*
Natural Language Processing and Knowledge Graph Lab,
School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, China
{guokaitang,zhaof}@hust.edu.cn

Abstract

Large language models (LLMs) show strong reasoning and decision-making ability, but their high inference cost motivates transferring agentic skills to small language models (SLMs). Agent distillation trains SLMs on full *reason–act–observe* trajectories from a tool-using teacher, enabling SLMs to acquire the tool-use capabilities of large teacher models. However, some teacher-agent trajectories are simply hard for the student to learn, and their compatibility with the student can vary widely; moreover, a uniform token-level loss prevents SLMs from learning the tool-use patterns and final decisions that truly drive successful reasoning. Therefore, we propose **SmartAD**, a **capacity-aligned agent distillation** framework that improves both the distilled data and the supervision signal. SmartAD (i) selects, for each training example, the trajectory with the minimum negative log-likelihood among multiple correct teacher samples to obtain student-friendly training data, and (ii) applies a *segment-weighted loss* that emphasizes *action execution* and *final decision* spans over intermediate reasoning. Experiments on multi-hop QA and math benchmarks with 1.5B and 3B models show that SmartAD consistently outperforms all baselines. Overall, our method enables small models to learn the teacher's capabilities more easily and efficiently through trajectory selection and segment-weighted supervision, achieving capacity-aligned distillation.

1 Introduction

Large language models (LLMs) have achieved strong performance across a wide range of complex reasoning and decision-making tasks (DeepSeek-AI et al., 2025; OpenAI, 2025). However, their high inference cost and substantial deployment footprint make large models hard to use at scale in scenarios with strict latency and cost constraints (Chen et al., 2025). This has motivated growing interest in small language models (SLMs), which offer lower inference cost and faster response. Yet, when tasks require long-horizon reasoning, tool usage, or multi-step interaction with the environment, SLMs still fall significantly short of LLMs (Shen et al., 2024).

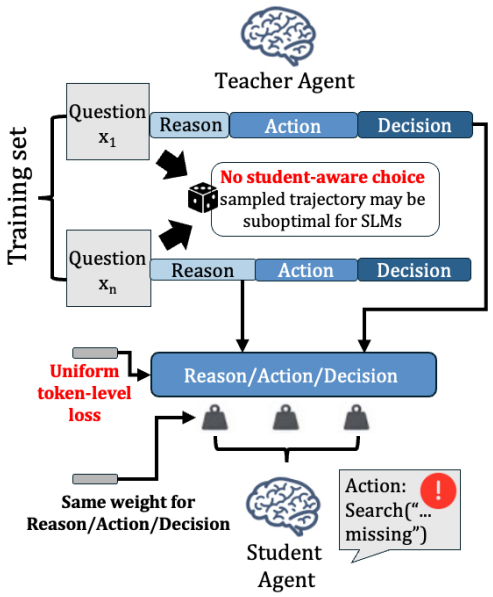

Figure 1: Current agent distillation methods are largely capacity-unaware: they neither select trajectories nor design supervision signals in a way that matches small-model capacity.

To further close this gap, a line of work explores reasoning distillation, where SLMs are trained to imitate chain-of-thought (CoT) traces generated by a larger teacher model via next-token prediction (Li et al., 2023). By exposing the student to explicit intermediate reasoning steps rather than only final answers, these methods aim to transfer richer problem-solving patterns from LLMs into much smaller models (Hsieh et al., 2023). However, directly distilling CoT traces into SLMs becomes fundamentally inadequate once problem solving

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 27045–27057

requires tool use, code execution, or precise numerical computation (Yin et al., 2025). CoT-only supervision can teach a student to produce fluent-looking explanations, but it does not tell the model when to call a tool, how to format and execute tool calls, or how to update its plan after observing tool outputs.

To address these issues, recent work has shifted from static CoT distillation to agent distillation, where a student model imitates the full reason, act, and observe trajectories of a LLM agent. By cloning the teacher's reasoning process and tool-augmented actions, an SLM can learn to call code, retrieval, or environment APIs, thereby generalizing better to queries that require new knowledge or computations. However, existing agent distillation frameworks, still process teacher trajectories in a coarse way: they either apply a uniform token level loss over the entire trajectory (Kang et al., 2025), or at best split it into "reasoning" and "action" spans with equal weights (Liu et al., 2025), without explicitly modeling which parts of the trajectory are more informative or more learnable for a small model. Moreover, they typically do not adapt the training data to the student's capacity, but instead treat all teacher trajectories as equally useful supervision information, even when they are noisy or far beyond what the student can reliably learn, so the student is forced to imitate all behaviors indiscriminately, which can limit stability, inference efficiency, and downstream generalization.

Therefore, we propose **SmartAD, a capacity-aligned agent distillation framework for small language models**, which explicitly addresses what to learn from teacher agents and how strongly to learn different parts of a trajectory. First, instead of naively distilling an arbitrary teacher trajectory, we generate multiple trajectories per example from a large agent and use the student's own negative log-likelihood (NLL), combined to automatically select a student-friendly trajectory—one that the student already assigns relatively high probability while still solving the task. This selection step performs a form of implicit curriculum and data cleaning, filtering out unnecessarily complex, noisy, or idiosyncratic trajectories that are misaligned with the student's capacity. Second, given a selected trajectory, we segment it into explanatory reasoning spans, concrete action spans and final decision spans, and introduce a segment-weighted loss that assigns progressively larger weights to more decision-critical parts of the trajectory. In this way, the student is encouraged to loosely align with the teacher's overall reasoning style, but is trained more aggressively on the action and decision segments that matter most for task success. To summarize, our key contributions are as follows:

- We introduce an **NLL-based trajectory selection strategy**, aligning the distilled data with the student's capacity.
- We propose a **segment-weighted agent loss** that moves beyond a uniform token-level loss to emphasize which parts of the teacher trajectory small models truly benefit from.
- We validate our method on multi-hop QA and math benchmarks, **showing improved task success and generalization over all baselines**.

2 Related Works

2.1 CoT Distillation

Early CoT distillation work centers on transferring teacher-generated traces to SLMs. Shridhar et al. (2023) pioneered using CoT traces generated by large models to fine-tune a student model. Li et al. (2023) proposed symbolic CoT distillation with symbolic intermediate-step supervision. Kang et al. (2023) proposed knowledge-augmented reasoning distillation for knowledge-intensive tasks, by combining external retrieval with CoTs. Building on these ideas, several works turn to compressing and refining CoT structure itself. TokenSkip (Xia et al., 2025) skips low-utility tokens to shorten CoTs while keeping accuracy almost unchanged. Yuyang et al. (2025) derived the optimal CoT length as a function of model capacity and task difficulty, showing that the number of reasoning steps should be aligned with the model's capability. Shangziqi et al. (2025) further enhanced the reasoning ability of smaller models by pruning inefficient reasoning steps from CoTs generated from LLMs. This trend of compressing distilled knowledge also extends to complex graph reasoning tasks (Liu et al., 2024). These studies show a shift from merely distilling teacher CoTs to optimizing and compressing them, while highlighting that trajectories from large models are not always well suited for SLMs.

2.2 Agent distillation

Compared with distilling static CoT alone, agent distillation further focuses on transferring the full

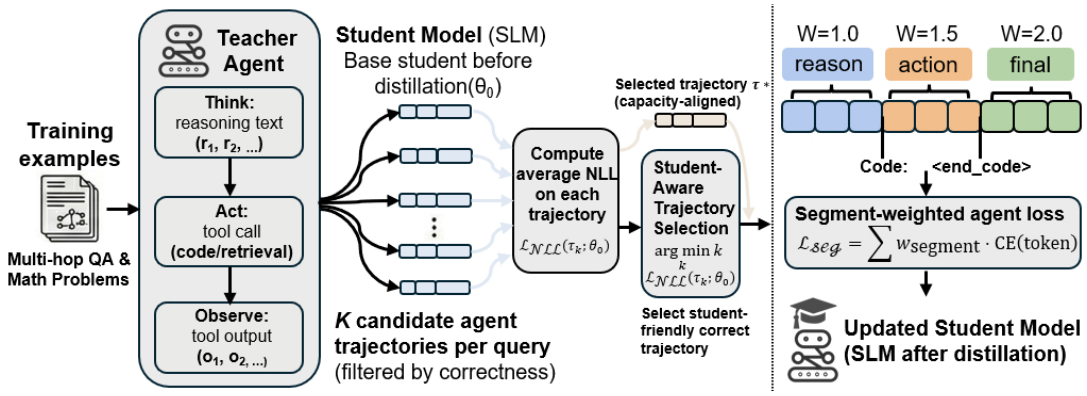

Figure 2: Our SmartAD framework: First, for each training example, we compute the negative log-likelihood (NLL) of all sampled trajectories under the student model and retain the trajectory with the lowest NLL. Second, during training, we apply a segment-weighted loss that assigns different weights to different parts of the trajectory.

reason–act–observe loop, so that the student not only learns how to "think," but also how to call tools and execute actions in the environment. For example, Kang et al. (2025) use an LLM agent with retrieval and code execution ability as the teacher, and distill teacher-trajectories, so that a small model learns to solve factual and mathematical tasks with the help of external tools. Structured Agent Distillation (Liu et al., 2025) explicitly splits an agent trajectory into several spans and applies span-specific objectives to each part, which helps alleviate the tendency of pure token level distillation to ignore the intrinsic connections between earlier and later steps. Agentdistill (Qiu et al., 2025) explores a training-free form of distillation based on MCP, using reusable tool boxes to transfer the tool-usage patterns of a large agent to the student. Chain-of-Agents (Li et al., 2025) compresses multi-agent collaboration into a base model in order to improve decision making and coordination on complex tasks. Although these methods substantially improve small models on tool-use and environment-interaction tasks, most of them do not explicitly select or restructure training trajectories according to the student's capacity, nor do they assign different weights to different parts of the trajectory when updating gradients.

3 Capacity-Aligned Agent Distillation Framework

3.1 Definition and Training Objective

Unlike CoT distillation, which supervises only static reasoning text, *agent distillation* teaches a student model to reproduce the structured decision-making behavior of a tool-using teacher agent. In this setting, a large model interacts with an external environment—such as a code interpreter, a retrieval API, or a web tool—by repeatedly **thinking**, **acting**, and **observing** the outcome of its actions. We refer to the resulting sequence as an *agent trajectory*.

Given an input query x, a teacher agent produces a trajectory composed of repeated reasoning–action–observation cycles:

$$\tau = \big((r_1, a_1, o_1), \ldots, (r_L, a_L, o_L)\big), \qquad (1)$$

where r_t denotes an intermediate reasoning step, a_t is a concrete tool-use action (e.g., a code snippet or a search command), and o_t is the environment's feedback returned in response to a_t. Importantly, o_t is not generated by the model but obtained from the external tool, making the trajectory grounded in real execution rather than text-only simulation. In our method, we adopt the formulation from CodeAct (Wang et al., 2024b), where each step consists of a Thought, Action, and Observation.

The goal of agent distillation is to fine-tune a small language model so that it can follow the same interactive procedure as the teacher. Following standard practice, only the *model-generated* parts of the trajectory—reasoning and actions—are included in the training loss, while the observations are treated as read-only context. Formally, the student minimizes:

$$\min_{\theta} \; \mathbb{E}_{x\sim D} \, \mathbb{E}_{\tau\sim T(x)} \sum_{t=1}^{L} -\log p_\theta(r_t, a_t \mid x, \tau_{<t}), \qquad (2)$$

where $\tau_{<t}$ includes all previous reasoning steps, actions, and environment observations. This objective encourages the student to learn not only *how*

to think, but also *when and how to invoke external tools*, and *how to adapt its next move after receiving tool feedback*. A distilled student trained on such trajectories can therefore perform tool-assisted reasoning, such as generating executable code, handling interpreter errors, refining search queries, or updating its plan based on retrieved evidence.

However, as discussed in Section 1 and 2, existing agent distillation frameworks often ignore two crucial factors:

- Not all trajectories are equally learnable for a small model—some are overly long, noisy, or stylistic, which can destabilize training; and
- Different segments of a trajectory contribute unequally to task success—action spans are harder and more critical than explanatory text, yet most methods apply a uniform distillation loss.

These limitations motivate our capacity-aligned agent distillation framework, which selects trajectories according to the student's ability and assigns differentiated training loss to different parts of the trajectory.

3.2 Student-Aware Trajectory Selection

For each training example x, we first sample K candidate trajectories from the teacher agent using a sampling temperature of 1.0:

$$\{\tau^{(k)}\}_{k=1}^{K} \sim p_T(\cdot \mid x, I_{\text{agent}}),$$

where in our experiments $K = 10$. Upstream filtering removes trajectories whose final answer is incorrect or whose tool calls fail, so all remaining candidates solve the task.

To measure how compatible a candidate trajectory is with a given student, we compute the macro-average of the negative log-likelihood (NLL) across its M assistant turns under the current student model. Let $\ell^{(m)}$ denote the mean per-token NLL for the m-th assistant response $y^{(m)}$. The final score is defined as:

$$\begin{aligned} \mathcal{L}_{\text{NLL}}(\tau;\theta) &= \frac{1}{M}\sum_{m=1}^{M} \ell^{(m)}, \\ \text{where } \ell^{(m)} &= \frac{1}{|y^{(m)}|}\sum_{y_j \in y^{(m)}} -\log p_\theta(y_j \mid h_{<j}). \end{aligned} \tag{3}$$

Here, $h_{<j}$ represents the full context history preceding token y_j. This hierarchical averaging ensures that each reasoning step contributes equally to the score, preventing longer turns from dominating the metric. A lower NLL means that the student already assigns higher probability to the teacher's next tokens, and that the gradient updates required to fit this trajectory are smaller and more stable. In implementation, we iterate through the trajectory turn-by-turn. For each assistant response, we construct the specific history context, compute its normalized loss $\ell^{(m)}$, and finally average these values to obtain $\mathcal{L}_{\text{NLL}}$.

For every input x, we then choose the most student-friendly trajectory:

$$k^\star(x) = \arg\min_{1\le k\le K} \mathcal{L}_{\text{NLL}}\big(\tau^{(k)};\theta_0\big), \tag{4}$$

where θ_0 denotes the parameters of the base student model before distillation. The resulting training set is $\tilde{\mathcal{D}} = \{(x, \tau^{(k^\star(x))})\}$.

Intuitively, trajectories with lower NLL lie closer to the student's current predictive distribution, and thus provide supervision that the student can absorb with fewer conflicting gradients. By always picking the lowest-NLL trajectory that is still correct, this selection step implicitly performs curriculum learning and data cleaning: it avoids forcing the student to imitate trajectories that are extremely unlikely under its current capacity (long, noisy, or idiosyncratic reasoning), and instead concentrates optimization on those trajectories that are both task-solving and student-aligned, which empirically leads to more stable training and better generalization.

3.3 Segment-Weighted Agent Loss

Even after trajectory selection, not all tokens in a trajectory are equally important for the student. Explanatory reasoning text, intermediate tool calls, and the final decision span differ in both their difficulty and their contribution to task success. In particular, due to limited capacity, small models often fail to reliably imitate the teacher's tool-use behaviors and final decisions, even if they can produce plausible reasoning text; as a result, action and final decision segments are more critical for successful execution. To reflect this, we introduce a segment-weighted variant of the agent distillation loss.

For a given input–trajectory pair (x, τ), let $\{y_i\}_{i=1}^{L}$ be the supervised token sequence after applying the instruction–response template and masking non-assistant tokens. We assign each token a segment label $s_i \in \{\text{reason}, \text{action}, \text{final}\}$.

In our data, the teacher agent wraps every tool invocation between string markers `"Code:"` and `"<end_code>"`. We therefore detect code blocks by searching for these markers in the token sequence and treat each contiguous block as the *action* segment, with the last block corresponding to the *final* segment that produces the final answer. All remaining supervised tokens (before any `"Code:"` marker or between blocks) are treated as *reason* segment.

Given a token-level cross-entropy

$$\ell_i(\theta) = -\log p_\theta(y_i \mid x, y_{<i}),$$

we introduce segment-dependent weights $w_{\text{reason}}, w_{\text{action}}, w_{\text{final}}$ and define the segment-weighted loss

$$\mathcal{L}_{\text{seg}}(x, \tau; \theta) = \frac{\sum_{i=1}^{L} w_{s_i}\, \ell_i(\theta)}{\sum_{i=1}^{L} w_{s_i}}, \qquad (5)$$

where w_{s_i} denotes the weight associated with the label s_i. In our experiments we use a simple monotone scheme: $w_{\text{reason}} = 1$, $w_{\text{action}} = 1.5$, and $w_{\text{final}} = 2$. Thus the model is still encouraged to loosely match the teacher's overall reasoning style, but it receives stronger gradients on the intermediate tool calls and, most importantly, on the final tool-augmented answer that directly determines task success.

Algorithmically, Eq. (5) is implemented by first computing a standard per-token cross-entropy with `ignore_index = -100`, then constructing a weight tensor matching the label positions, and finally averaging the losses with weights w_{s_i}. This objective is plugged into the SFT trainer as a drop-in replacement for the uniform token-level loss.

Overall, Our method SmartAD combines student-aware trajectory selection (Eq. 4) with the segment-weighted loss (Eq. 5) on the selected trajectories, yielding a capacity-aligned agent distillation procedure that better matches what small models can realistically learn from large teacher agents.

4 Experiments

4.1 Experimental setup

Tasks and Benchmarks. We evaluate our method on two classes of reasoning tasks: factual multi-hop reasoning and mathematical problem solving, with the goal of assessing both in-domain performance and out-of-domain generalization of distilled SLMs. We use 1000 HotPotQA and 2000 MATH examples for training, following Kang et al. (2025).

Domain	Dataset	Task
In	HotPotQA (Yang et al., 2018)	2-hop QA
OOD	Bamboogle (Press et al., 2023)	2-hop QA
OOD	MuSiQue (Trivedi et al., 2022)	3-hop QA
OOD	2WikiQA (Ho et al., 2020)	2-hop QA
In	Math500 (Hendrycks et al., 2021)	College-level
OOD	Gaokao (Zhong et al., 2024)	High school-level
OOD	AMC23	Competition-level
OOD	AIME (Aime, 2024)	Olympiad-level
OOD	OlymMath (Sun et al., 2025)	Olympiad-level

Table 1: Evaluation benchmarks. In denotes in-domain and OOD denotes out-of-domain. AIME consists of math problems from the years 2022 to 2024. QA datasets are constructed following Kang et al. (2025)

Baselines. We compare our method against seven baselines spanning prompting and distillation. **CoT Prompting** prompts the student to produce a CoT without training; **CoT Distillation** fine-tunes the student on teacher-generated CoT traces; **CoT Distillation + RAG** adds retrieval function; **Agent Prompting** prompts the student to follow a reason–act–observe format at test time without training; **Agent Distillation** fine-tunes on correctness-filtered teacher trajectories with a uniform token-level loss (excluding observation tokens); **SAD (Training only)** uses the segment-aware loss of Liu et al. (2025); **AD (Training only)** uses the first-thought prefix strategy of Kang et al. (2025).

Training & inference details. We use Qwen2.5-32B-Instruct as the teacher model and Qwen2.5-3B-Instruct and Qwen2.5-1.5B-Instruct as student models, distilling trajectories from Qwen2.5-32B-Instruct to train the student models. We used LoRA (Hu et al., 2022) to fine-tune the student models, training each student model for two epochs. See Appendix A for detailed training details. In the inference process, we set the temperature to 0.0 and the maximum number of steps the agent can execute to 5. We use e5-base-v2 (Wang et al., 2024a) for semantic retrieval and employ the Wikipedia 2018 as the shared knowledge base for both agents and the RAG.

4.2 Main results

Table 2 reports the pass@1 accuracy of different baselines and our method on both in-domain and out-of-domain benchmarks, using 1.5B and 3B students. Overall, **SmartAD consistently achieves**

Params	Type	Method	In-domain		Out-of-domain							Avg
			HotPotQA	Math500	MuSiQue	Bamboogle	2WikiQA	AIME	AMC23	Gaokao	OlymMath	
1.5B	CoT	Prompt	19.6	**48.2**	**4.4**	25.6	18.6	**3.3**	**27.5**	**38.18**	2.5	20.88
		Distill	20.2	44.2	3.0	23.2	20.6	**3.3**	20.0	35.84	**3.5**	19.32
		Distill + RAG	**41.8**	43.8	4.0	**32.0**	**31.2**	1.1	15.0	31.68	2.0	22.51
	Agent	Prompt	21.2	26.2	2.6	12.0	24.8	**5.5**	**25.0**	20.00	2.0	15.48
		Distill	44.0	40.6	8.2	29.6	38.2	3.3	22.5	28.31	3.0	24.19
		AD	44.6	41.0	11.4	38.4	40.2	2.2	22.5	29.35	2.5	25.8
		SAD	46.8	39.0	7.8	30.4	35.8	3.3	20.0	27.01	2.0	23.58
		SmartAD (ours)	**47.4**	**44.4**	**12.8**	**44.0**	**40.6**	3.3	**25.0**	**34.28**	**4.0**	**28.42**
3B	CoT	Prompt	25.0	**61.4**	5.0	33.6	21.8	4.4	**47.5**	**46.75**	**5.0**	27.83
		Distill	28.0	57.4	5.0	35.2	23.8	**8.9**	35.0	43.63	4.5	26.83
		Distill + RAG	**45.2**	54.8	**6.2**	**40.8**	**36.8**	4.4	25.0	41.03	3.5	28.64
	Agent	Prompt	42.8	32.0	10.0	32.0	30.4	3.3	27.5	20.25	2.0	22.25
		Distill	50.6	52.8	15.0	43.2	41.4	4.4	30.0	36.88	4.0	30.92
		AD	50.4	**57.0**	15.4	45.6	43.2	**7.7**	30.0	**42.59**	4.5	32.9
		SAD	49.6	48.6	13.6	38.4	**44.6**	2.2	22.5	33.76	3.5	28.53
		SmartAD (ours)	**53.4**	55.0	**17.0**	**50.4**	**44.6**	**7.7**	**32.5**	38.70	**6.5**	**34.0**

Table 2: Pass@1 accuracy on in-domain and out-of-domain benchmarks. For accuracy evaluation, we apply exact-match scoring for math datasets, whereas for factual reasoning tasks, we employ an LLM-as-a-judge paradigm with Qwen-Plus as the evaluator. **Bold numbers** indicate the best results, and underlined numbers indicate the second-best results.

the best average performance across the two model scales, indicating that aligning training trajectories and supervision strength with student capacity yields more effective agent distillation.

SmartAD outperforms strong agent distillation baselines. For the **1.5B** student, SmartAD reaches an average score of **28.42**, improving over vanilla agent distillation (**24.19**) by **+4.23** points and over the AD baseline (**25.80**) by **+2.62** points. Gains are particularly clear on out-of-domain factual benchmarks, e.g., MuSiQue (12.8 vs. 11.4), Bamboogle (44.0 vs. 38.4), and on math generalization benchmarks such as OlymMath (4.0 vs. 2.5). These results suggest that SmartAD is more robust to different domains and better at transferring actionable tool-use behaviors into smaller students. For the **3B** student, SmartAD again achieves the best average score (**34.00**), outperforming vanilla agent distillation (**30.92**) by **+3.08** points and the AD baseline (**32.90**) by **+1.10** points. Improvements are most pronounced on out-of-domain multi-hop QA (e.g., Bamboogle: 50.4 vs. 45.6) and harder math benchmarks (e.g., OlymMath: 6.5 vs. 4.5), while remaining competitive on in-domain Math500 (55.0 vs. 57.0).

Our method better matches the goal of capacity-aligned tool use. Across both scales, CoT prompting and CoT distillation are consistently less effective than agent-based training, especially on benchmarks that require tool invocation, code execution, or multi-step interaction. Even when CoT distillation is augmented with retrieval, it still lacks explicit supervision over *how* to act with tools and *how* to revise decisions based on observations, which are central to agentic problem solving. In contrast, agent distillation transfers full reason–act–observe trajectories and therefore provides direct training signals for actionable behavior. This gap aligns with our core motivation: to endow small models with agentic capabilities under limited capacity, it is more crucial to distill interactive trajectories and to align both data and supervision with what the student can reliably learn, rather than to imitate static CoT traces alone.

Scaling trend. Moving from 1.5B to 3B generally improves performance for all methods, but SmartAD maintains its advantage at both scales. This suggests that the proposed capacity-aligned design is not tied to a particular model size and can serve as a general recipe for stabilizing and strengthening small-model agent distillation.

4.3 Ablation Study

Table 3 reports an ablation study on two student scales (1.5B and 3B) to isolate the contributions of our two core components: student-aware trajectory selection via NLL and segment-weighted agent distillation loss.

NLL-based selection provides capacity-aligned supervision. Adding NLL trajectory selection consistently improves the overall average accuracy over vanilla agent distillation for both

Params	Variant	In-domain		Out-of-domain							Avg
		HotPotQA	Math500	MuSiQue	Bamboogle	2WikiQA	AIME	AMC23	Gaokao	OlymMath	Avg
1.5B	Agent Distillation	44.0	40.6	8.2	29.6	38.2	3.3	22.5	28.31	3.0	24.19
	+ NLL Trajectory Selection	46.2	**45.6**	7.8	40.0	35.4	**5.6**	15.0	31.42	3.5	25.60
	+ Segment-Weighted Loss	47.0	42.2	10.4	38.4	**43.6**	**5.6**	**25.0**	30.38	3.0	27.30
	SmartAD	**47.4**	44.4	**12.8**	**44.0**	40.6	3.3	**25.0**	**34.28**	**4.0**	**28.42**
3B	Agent Distillation	50.6	52.8	15.0	43.2	41.4	4.4	**30.0**	36.88	4.0	30.92
	+ NLL Trajectory Selection	51.0	53.4	16.4	**52.0**	44.2	5.5	30.0	37.10	4.5	32.68
	+ Segment-Weighted Loss	50.8	52.0	13.4	40.8	**47.6**	4.4	30.0	38.18	3.0	31.13
	SmartAD	**53.4**	**55.0**	**17.0**	50.4	44.6	**7.7**	**32.5**	**38.70**	**6.5**	**34.00**

Table 3: Ablation results of SmartAD components.

Benchmark	min-NLL	medium-NLL	max-NLL
2WikiQA	**44.2**	41.0	43.0
AIME	**5.5**	4.4	3.3
AMC23	**30.0**	**30.0**	**30.0**
Bamboogle	**52.0**	46.4	45.6
Gaokao	37.1	**37.5**	37.0
HotPotQA	**51.0**	48.4	49.0
Math500	**53.4**	52.8	52.8
MuSiQue	**16.4**	13.6	14.0
OlymMath	4.5	4.5	**5.5**
Avg	**32.68**	30.95	31.13

Table 4: Comparison of different NLL-based trajectory selection criteria on the 3B student at $K = 10$ under the standard uniform token-level loss.

model sizes. For the 1.5B student, "+ NLL Trajectory Selection" increases the average score from 24.19 to 25.60 (+1.41), indicating that selecting the minimum-NLL trajectory makes the training signal more learnable for the student model. For the 3B student, we observe a similar trend, where the average improves from 30.92 to 32.68 (+1.76). These results support our capacity-alignment hypothesis: among multiple correct teacher trajectories, the one that is most probable under the base student acts as a *student-friendly* demonstration that reduces distribution mismatch and implicitly implements a curriculum learning over trajectories.

Segment-weighted loss strengthens learning of action execution and decision making. Applying the segment-weighted loss alone also brings a notable gain, especially for the 1.5B student (24.19 $\rightarrow$ 27.30, +3.11). This suggests that uniform token-level supervision under-utilizes agent trajectories, because not all parts of a trajectory contribute equally to final task success. By reweighting spans to emphasize action execution and final decision regions, the student receives stronger gradients on the behaviors that directly determine correctness, leading to improved generalization across both factual and mathematical benchmarks.

Combining both yields the best and most stable performance. Our full method, **SmartAD**, achieves the highest average score for both model sizes: 28.42 for 1.5B and 34.00 for 3B. Compared to the base agent distillation, SmartAD improves the average by +4.23 (1.5B) and +3.08 (3B). This consistent improvement indicates that the two components are complementary: NLL selection aligns the *data distribution* with the student's capacity, while segment-weighted loss aligns the *optimization signal* with the importance of different trajectory regions.

4.4 Alternative NLL Selection Criteria

One potential concern is that selecting the minimum-NLL trajectory may bias training toward overly easy trajectories. To examine this, we compare three selection strategies on the 3B student with $K = 10$: **min-NLL**, **medium-NLL**, and **max-NLL**, where all candidates are correct teacher trajectories for the same input. Here, medium-NLL selects the correct trajectory with the median student NLL, and max-NLL selects the correct trajectory with the highest student NLL. Table 4 shows that **min-NLL** achieves the best overall average performance (32.68), outperforming both **medium-NLL** (30.95) and **max-NLL** (31.13). The gains are especially clear on Bamboogle, HotPotQA, and MuSiQue.

These results suggest that the trajectory selected by min-NLL is not simply "easy" in an absolute sense, but rather better aligned with the current student's learnable distribution. In contrast, forcing the student to imitate higher-NLL trajectories tends to introduce stronger distribution mismatch and optimization instability, whose cost outweighs the potential benefits under the current student capacity.

Benchmark	SmartAD	Uniform	ThoughtH.	ActionH.
HotPotQA	**47.4**	46.6	45.6	45.2
Math500	**44.4**	42.6	44.0	**44.4**
MuSiQue	**12.8**	11.2	11.2	11.6
Bamboogle	**44.0**	39.2	39.2	40.0
2WikiQA	40.6	**43.2**	42.4	38.6
AIME	3.3	3.3	**4.4**	**4.4**
AMC23	**25.0**	17.5	22.5	17.5
Gaokao	**34.28**	32.72	33.24	32.4
OlymMath	4.0	**4.5**	2.5	3.5
Avg	**28.42**	26.80	27.20	26.40

Table 5: Effect of different segment-weighted objectives on the 1.5B student. Uniform uses weights 1:1:1 for reasoning, action, and final decision. ThoughtHeavy uses 2:1:1 and ActionHeavy uses 1:2:1.

Benchmark	$K=1$	$K=5$	$K=10$	$K=15$
2WikiQA	41.4	46.4	44.2	**48.2**
AIME	4.4	5.5	5.5	**6.6**
AMC23	**30.0**	22.5	**30.0**	**30.0**
Bamboogle	43.2	42.4	**52.0**	46.4
Gaokao	36.88	34.2	**37.1**	36.1
HotPotQA	50.6	49.8	51.0	**51.4**
Math500	52.8	51.4	**53.4**	51.4
MuSiQue	15.0	13.8	**16.4**	16.2
OlymMath	4.0	5.5	4.5	**6.0**
Avg	30.92	30.16	**32.68**	32.47

Table 6: Effect of the number of sampled teacher trajectories K on the 3B student under the uniform token-level loss.

4.5 Comparative Experiment

Table 5 compares four training objectives based on NLL trajectory selection, using LoRA fine-tuning on Qwen2.5-1.5B-Instruct. Overall, our proposed **SmartAD** achieves the best average performance (**28.42**), outperforming **Uniform** (26.80), **ThoughtHeavy** (27.20), and **ActionHeavy** (26.40). These results suggest that emphasizing the segments directly related to *action execution* and *final decision* helps small models better internalize tool-augmented behaviors. In contrast, emphasizing only reasoning tokens or only action tokens is less effective than jointly emphasizing both action and final-decision segments.

In particular, **ActionHeavy** (weights 1:2:1) performs worse than SmartAD on most benchmarks and yields the lowest average score among the segment-reweighted variants. Together with the weaker performance of **ThoughtHeavy** (weights 2:1:1), this indicates that simply allocating more weight to a single segment is insufficient. For tool-using agents, a more balanced emphasis on *action execution* and *final decision making*, as in SmartAD, is generally more beneficial for overall task success.

4.6 Effect of K and Computational Overhead

SmartAD introduces additional offline cost compared with standard agent distillation, since for each training example we sample multiple teacher trajectories and compute the student-aware negative log-likelihood (NLL) to select the most compatible one. However, this overhead is incurred only once during data construction, and does not affect either student training or test-time inference. In practice, the overhead is modest: under our compute-provider pricing, filtering 3,000 training instances with $K = 10$ costs approximately 40 RMB. If the teacher model is deployed locally, the additional monetary cost is close to zero. We therefore view this as a favorable trade-off: SmartAD spends a small amount of extra computation during offline data preparation to obtain higher-quality and more student-aligned supervision.

We further study the sensitivity of SmartAD to the number of sampled trajectories K on the 3B student model, considering $K \in \{1, 5, 10, 15\}$. The results are shown in Table 6. Overall, the average performance peaks at $K = 10$, improving from 30.92 at $K = 1$ to 32.68. Increasing K further to 15 does not bring additional average gains, with the overall score slightly dropping to 32.47, although a few individual benchmarks still improve. This suggests that using a moderate number of candidate trajectories is sufficient to capture most of the benefit of student-aware selection, and that $K = 10$ provides the best trade-off between effectiveness and cost in our setting.

4.7 Analysis Study

To validate that our NLL-based trajectory selection is well motivated, for each training set, we compute per-trajectory negative log-likelihood (NLL) according to 1.5B model on teacher-generated agent trajectories, and partition trajectories into three tertiles (EASY/MEDIUM/HARD) by NLL. We report the accuracy (task success rate) together with the corresponding mean NLL in Fig. 3. We consider two bucketing strategies: **global**, where all trajectories are pooled and split into EASY/MEDIUM/HARD by global NLL quantiles, and **per-question**, where trajectories for each question are sorted and split

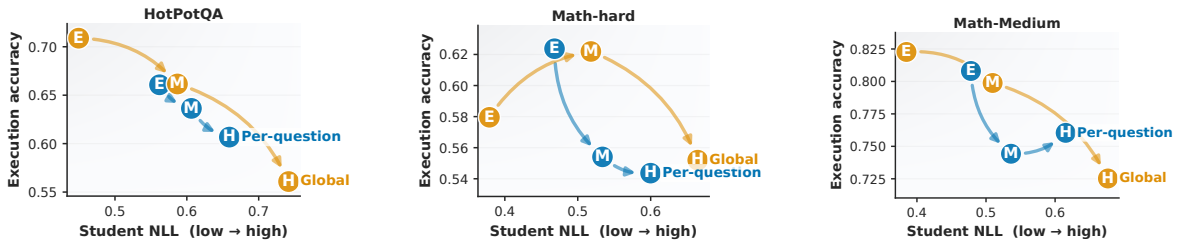

Figure 3: We sort sampled trajectories by student NLL (ascending) into three contiguous buckets—E:EASY (lowest), M:MEDIUM (middle), and H:HARD (highest)—and report mean NLL and accuracy per bucket. "Math-hard" and "Math-medium" denote MATH problems of difficulty levels 4–5 and 2–3, respectively.

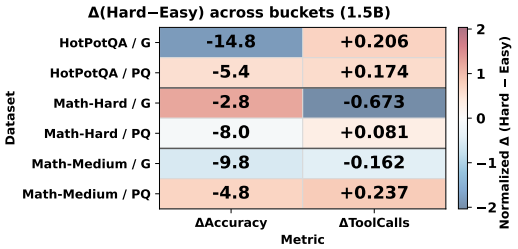

Figure 4: Δ(Hard–Easy) heatmap for accuracy and tool calls across datasets.

independently by their within-question NLL quantiles.

Under **per-question** bucketing, we consistently observe a negative association between student NLL and execution accuracy across HOTPOTQA, MATH-HARD, and MATH-MEDIUM: trajectories assigned lower NLL by the 1.5B student are more likely to be executed correctly, even after controlling for question-specific effects. This is exactly the regime our method operates in—For each question, we generate multiple teacher trajectories and simply pick the one with the lowest student NLL; the results show that lower NLL usually means the trajectory is easier for the student to learn and execute correctly. Turning to **global** bucketing, the same pattern is clear on HOTPOTQA and MATH-MEDIUM: as the NLL value of the trajectory gradually increases, the accuracy decreases. However, for MATH-HARD, the accuracy rate for the easy bucket was slightly lower than that for the medium bucket. We attribute this to the fact that, for harder problems, trajectories the student deems "easier" (i.e., with lower NLL) often correspond to shorter or more heuristic behaviors; however, such behaviors are not necessarily sufficient to reach a correct solution. Importantly, this exception further confirms our design choice: rather than relying on a *global* notion of difficulty, we perform *per-question* selection and always choose the minimum-NLL trajectory within the candidate set, which avoids comparing NLL across different questions and instead picks the easiest-to-learn trajectory among candidates for the same question. Moreover, under *per-question* setting, lower-NLL buckets also tend to have fewer tool calls as shown in Fig. 4, suggesting that student NLL correlates with more concise and operationally simpler trajectories that are easier for a small model to imitate and execute. See Appendix B for details.

Overall, the analysis shows that students' NLL reliably reflects trajectory difficulty: lower-NLL trajectories are easier for small models to learn, so selecting the minimum-NLL trajectory provides supervision that better matches the student's capacity and delivers more capacity-aligned training signals.

5 Conclusion

We presented **SmartAD**, a capacity-aligned agent distillation framework for transferring the capabilities of tool-using LLM teachers to smaller language models. SmartAD explicitly considers *what* small models should learn and *how strongly* different parts of the trajectory should be learned. Specifically, our method combines student-aware trajectory selection, which chooses the most learnable correct trajectory according to the student's own NLL, with a segment-weighted training objective that places greater emphasis on action execution and final decision spans.

Extensive experiments on multi-hop QA and mathematical reasoning benchmarks show that SmartAD consistently outperforms strong prompting and distillation baselines across both 1.5B and 3B students, while also generalizing well to out-of-domain settings. Overall, these findings highlight the importance of capacity-aware data selection and supervision design for building stronger and more efficient small model agents.

Limitations

Our work has several limitations. First, SmartAD requires sampling multiple teacher trajectories per example and computing student negative log-likelihood for selection, which increases offline data generation and preprocessing cost compared to standard agent distillation. Second, we focus on improving training-time data and supervision, and do not incorporate test-time scaling mechanisms (e.g., increased inference-time search, self-consistency, or additional tool-calling budget), which could further boost performance but is outside the scope of this work.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2023YFF0905503, National Natural Science Foundation of China under Grants No.62472188.

References

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

Xiao Chen, Changyi Ma, Wenqi Fan, Zhaoxiang Zhang, and Li Qing. 2025. C2KD: Cross-layer and cross-head knowledge distillation for small language model-based recommendation. In *Findings of the 63th Annual Meeting of the Association for Computational Linguistics*, pages 17827–17838, Vienna, Austria. Association for Computational Linguistics.

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, and 181 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Preprint*, arXiv:2501.12948.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, volume 1, Virtual. Curran Associates, Inc.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*, Virtual. OpenReview.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. 2025. Distilling LLM agent into small models with retrieval and code tools. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*, San Diego, USA. Curran Associates, Inc.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, volume 36, pages 48573–48602, New Orleans, USA. Curran Associates, Inc.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2665–2679, Toronto, Canada. Association for Computational Linguistics.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, and 11 others. 2025. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *Preprint*, arXiv:2508.13167.

Jun Liu, Zhenglun Kong, Peiyan Dong, Changdi Yang, Tianqi Li, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Pu Zhao, Xue Lin, Dong Huang, and Yanzhi Wang. 2025. Structured agent distillation for large language model. *Preprint*, arXiv:2505.13820.

Kangzheng Liu, Feng Zhao, Yu Yang, and Guandong Xu. 2024. Dysarl: Dynamic structure-aware representation learning for multimodal knowledge graph reasoning. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8247–8256, Melbourne, Australia. ACM.

OpenAI. 2025. Introducing OpenAI O3 and O4-Mini. https://openai.com/index/introducing-o3-and-o4-mini.

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

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, Shilong Liu, Xun Jiang, Liu Leqi, and Mengdi Wang. 2025. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *Preprint*, arXiv:2506.14728.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. Small LLMs are weak tool learners: A multi-LLM agent. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16658–16680, Miami, Florida, USA. Association for Computational Linguistics.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 7059–7073, Toronto, Canada. Association for Computational Linguistics.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024a. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in LLMs. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*, Singapore EXPO. OpenReview.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3351–3363, Suzhou, China. Association for Computational Linguistics.

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

Fan Yin, Zifeng Wang, I-Hung Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long Le, Kai-Wei Chang, Chen-Yu Lee, Hamid Palangi, and Tomas Pfister. 2025. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 32600–32616, Vienna, Austria. Association for Computational Linguistics.

Shangziqi Zhao, Jiahao Yuan, Guisong Yang, and Usman Naseem. 2025. Can pruning improve reasoning? revisiting long-cot compression with capability in mind for better reasoning. *Preprint*, arXiv:2505.14582.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

Appendix

Metric	G-E	G-M	G-H	PQ-E	PQ-M	PQ-H
N_{traj}	3181	3181	3181	5116	2190	2237
N_{q}	586	699	625	997	712	851
Mean NLL	0.385	0.510	0.676	0.478	0.536	0.615
Traj. chars	1232.7	947.4	897.1	965.2	1098.0	1093.3
Tool calls	2.340	2.199	2.178	2.134	2.350	2.371
Acc. (%)	82.3	79.9	72.5	80.8	74.4	76.0
Q words	24.4	25.9	28.9	25.2	28.6	27.1

Table 7: Bucket-level statistics for MATH-MEDIUM trajectories under a 1.5B student. Columns are tertiles: Global (G) and Per-question (PQ). E, M, and H represent easy, medium, and hard respectively. Q words represents the average question length.

Appendix A

We use QWEN2.5-32B-INSTRUCT as the teacher model and QWEN2.5-3B-INSTRUCT and QWEN2.5-1.5B-INSTRUCT as student models. For each dataset, we fine-tune the student with supervised fine-tuning (LoRA) on teacher-generated agent trajectories after correctness filtering.

Training is implemented with `torchrun` (`-nproc_per_node=2`). Unless otherwise specified, we train for 2 epochs, using a per-device batch size of 1 and gradient accumulation of 2 steps, resulting in a global effective batch size of 4. We use AdamW with a learning rate of 2×10^{-4}. The maximum sequence length is set to 10,240 tokens (`-max_length 10240`). To reduce memory usage for long trajectories, we enable gradient checkpointing (`-gradient_checkpointing`) and use DeepSpeed ZeRO-0 (`ds_z0_config.json`).

All experiments were conducted on 2 RTX PRO 6000 (96GB) GPUs and all agent baselines follow the same training settings.

Appendix B

The specific meanings of global bucketing and per_question bucketing. Global bucketing. We pool the NLL values of all trajectories in the entire dataset, sort them globally, and split them into EASY/MEDIUM/HARD according to global quantiles (e.g., tertiles). **Per-question bucketing.** For each question, we consider only its multiple candidate trajectories, sort them by NLL, and split them into EASY/MEDIUM/HARD using quantiles computed *within that question*, so that difficulty is assessed by comparing trajectories generated for the same question only.

Metric	G-E	G-M	G-H	PQ-E	PQ-M	PQ-H
N_{traj}	3092	3084	3087	4422	2330	2511
N_{q}	638	759	648	995	818	904
Mean NLL	0.379	0.518	0.664	0.468	0.534	0.600
Traj. chars	2212.1	1337.0	1207.3	1588.7	1599.2	1568.5
Tool calls	3.021	2.475	2.348	2.575	2.645	2.656
Acc. (%)	58.0	62.2	55.2	62.4	55.4	54.4
Q words	38.2	38.1	41.1	38.0	40.0	40.3

Table 8: Bucket-level statistics for MATH-HARD trajectories under a 1.5B student. Columns are tertiles: Global (G) and Per-question (PQ). E, M, and H represent easy, medium, and hard respectively. Q words represents the average question length.

Metric	G-E	G-M	G-H	PQ-E	PQ-M	PQ-H
N_{traj}	3335	3336	3329	5659	2200	2141
N_{q}	501	603	535	1000	675	803
Mean NLL	0.449	0.587	0.742	0.562	0.606	0.659
Traj. chars	1120.1	1070.6	1127.6	1072.1	1124.5	1176.9
Tool calls	3.078	3.224	3.284	3.131	3.254	3.305
Acc. (%)	70.9	66.2	56.1	66.1	63.6	60.7
Q words	15.6	15.8	15.8	15.6	15.8	16.0

Table 9: Bucket-level statistics for HOTPOTQA trajectories under a 1.5B student. Columns are tertiles: Global (G) and Per-question (PQ). E, M, and H represent easy, medium, and hard respectively. Q words represents the average question length.

Trajectory complexity vs. student NLL. In addition to execution success, student NLL also correlates with the *operational complexity* of teacher trajectories, as reflected by trajectory length (measured in characters) and the number of tool calls. Under **per-question** bucketing, lower-NLL trajectories are generally more concise and require fewer tool invocations. For HOTPOTQA (Table 9), the easy bucket has shorter trajectories (1072 vs. 1177 chars) and fewer tool calls (3.131 vs. 3.305) than the hard bucket. A similar pattern holds on MATH-MEDIUM (Table 7), where the easy bucket is substantially shorter (965 vs. 1093 chars) and uses fewer tools (2.134 vs. 2.371) than the hard bucket. For MATH-HARD (Table 8), trajectory length varies only mildly across buckets (about 1569–1599 chars), but tool calls still increase with NLL (2.575 $\rightarrow$ 2.656), indicating that low-NLL trajectories tend to be slightly less tool-intensive even on harder problems. Notably, the average question length is similar across buckets within each dataset (e.g., 15.6–16.0 words for HOTPOTQA and 38.0–40.3 words for MATH-HARD), suggesting that these differences are not simply driven by longer questions.

Method	HumanEval (Accuracy, %)
Qwen2.5-3B-Instruct	45.06
Agent Distillation	49.67
AD	51.97
SAD	43.75
SmartAD (Ours)	**52.34**

Table 10: Results on HumanEval for the 3B student model. SmartAD achieves the best code-generation performance among all compared methods.

In contrast, under **global** bucketing, the relationship between NLL and trajectory complexity can be weaker or even inverted, especially on math datasets. For example, in MATH-HARD (Table 8), the global easy bucket (lowest NLL) has much longer trajectories (2212 chars) and more tool calls (3.021) than the hard bucket (1207 chars; 2.348 calls). This indicates that comparing NLL across different questions can mix question difficulty and trajectory style: globally "easy" trajectories may come from easier questions that admit longer, more procedural tool usage, while higher-NLL trajectories may come from harder questions where the teacher produces shorter but harder-to-imitate behaviors.

Appendix C: Additional Results on HumanEval

To further evaluate the robustness and generalization of SmartAD beyond QA and mathematical reasoning tasks, we conduct an additional experiment on **HumanEval**, a standard benchmark for code generation. This evaluation directly measures whether the proposed capacity-aligned agent distillation strategy can also benefit pure programming ability.

Table 10 reports the results on Qwen2.5-3B-Instruct and several distilled variants. The base model Qwen2.5-3B-Instruct achieves 45.06 accuracy on HumanEval. Standard **Agent Distillation** improves the performance to 49.67, while **AD** reaches 51.97. Our proposed **SmartAD** achieves the best result, reaching **52.34**, outperforming all baselines.

These results suggest that SmartAD generalizes beyond the original multi-hop QA and math settings, and can also improve performance on pure code-generation tasks.