Towards Widening The Distillation Bottleneck for Reasoning Models

Anonymous ACL submission

Abstract

Large Reasoning Models (LRMs) such as OpenAI o1 and DeepSeek-R1 have shown remarkable reasoning capabilities by scaling test-time compute and generating long Chain-of-Thought (CoT). Distillation—post-training on LRMs-generated data—is a straightforward yet effective method to enhance the reasoning abilities of smaller models, but faces a critical bottleneck: we found that distilled long CoT data poses *learning difficulty* for small models and leads to the *inheritance of biases* (i.e., formalistic long-time thinking) when using Supervised Fine-tuning (SFT) and Reinforcement Learning (RL) methods. To alleviate this bottleneck, we propose constructing data from scratch using Monte Carlo Tree Search (MCTS). We then exploit a set of CoT-aware approaches, including *Thoughts Length Balance, Fine-grained DPO*, and *Joint Post-training Objective*, to enhance SFT and RL on the MCTS data. We conducted evaluation on various benchmarks such as math (GSM8K, MATH, AIME). instruction-following (Multi-IF) and planning (Blocksworld), results demonstrate our CoT-aware approaches substantially improve the reasoning performance of distilled models compared to standard distilled models via reducing the hallucinations in long-time thinking.

1 Introduction

Recent advancements in large reasoning models (LRMs), such as OpenAI o1 (OpenAI, 2024), QwQ (Qwen Team, 2024) and DeepSeek-R1 (Guo et al., 2025), have led to significant progress in handling complex tasks spanning mathematics, coding, and even open-ended queries (Zhong et al., 2024; Huang et al., 2024; Zhao et al., 2024). The success is largely attributed to "scaling test-time compute" by extending the length of the reasoning process. Given that most state-of-the-art LRMs are computationally expensive, recent efforts attempt to distill their reasoning capabilities into smaller lightweight models, demonstrating competitive performances (Qin et al., 2024). For instance, Guo et al. (2025) explored *direct distillation*, where they fine-tuned smaller dense models (e.g. Qwen2.5 7B) using reasoning patterns generated by DeepSeek-R1 671B model, outperforming GPT-4 on math benchmarks (e.g. AIME: 9.3% vs. 55.5%).

Error Type	Thoughts
Content Repetition (Math)	... So I've got this probability ... that the *positions* are considered up to consider that the *positions* are considered up to ... (no answer)
Over-Reflection (Planning)	So I have this problem: I need to ... Wait, *perhaps* it's ... Wait, *perhaps* I need to think differently ... I recall that ... (no answer)
Instruction Failure (Translation)	Translation: *Your order is closed.* Your order is closed. Wait, ... the same sentence twice. I should remove ... *Your order is closed.* Your order is closed. ... (no answer)

Table 1: The illustration of **formalistic long-time thinking** generated by distilled reasoning models across different tasks. Error tokens in thoughts are highlighted in *blue* and red colors. Notably, due to excessively long thoughts, there are *no final answers* in above cases. The Quantitative Analysis is detailed in Section 2.4.

However, we observed that these distilled models often exhibit hallucinations during long-time thinking, such as content repetition and over-reflection, leading to no final answer being produced (as shown in Table 1). We refer to this phenomenon as **formalistic long-time thinking**, where smaller models mechanically replicate the reasoning patterns of large models without internalizing the reasoning logic. Recent research shows that LRMs face both over-thinking and under-thinking issues (Chen et al., 2024; Wang et al., 2025), while smaller models struggle to learn gen-

eral reasoning (Fu et al., 2023). Accordingly, the root cause may be that distillation methods introduce *bias inheritance* and *learning difficulties* in smaller models. A natural research question arise: *How can long CoT reasoning be effectively transferred to smaller models through data construction, SFT and RL methods?*

To tackle this challenge, we explore improvements in reasoning distillation from both data and methodological perspectives. First, we propose a fundamental framework for constructing tree-based CoT data, which generates pre-defined thought nodes using general LLMs (rather than LRMs) and heuristically expands these nodes into a tree structure via the Monte Carlo Tree Search (MCTS) algorithm (Browne et al., 2012). The constructed data is not only more effective compared to directly distilled data, but also inherently more flexible, allowing the extraction of different types of reasoning paths as training data. Secondly, regarding commonly-used SFT and direct preference optimization (DPO) (Rafailov et al., 2023) as post-training framework, we empirically investigate a set of CoT-aware methods on the effects of formalistic long-time thinking. Specifically, this includes: 1) *Thoughts Length Balance*, where we extract CoT data of varying lengths; 2) *Fine-grained DPO*, where we employ conservative DPO (cDPO) (Mitchell, 2023) and mask-based DPO to better leverage the fine-grained information in long CoT; 3) *Joint Post-training Objective*, where we combine the DPO loss with SFT loss to mitigate the over-optimization observed in DPO (Fernando et al., 2024; Wang et al., 2024).

We validated our approaches on five exam-oriented and open-ended benchmarks, covering three different difficulty levels of math (GSM8K, MATH and AIME) (Cobbe et al., 2021a; Lightman et al., 2023), instruction-following in eight languages (Multi-IF) (He et al., 2024), and real-world planning tasks (Blocksworld) (Valmeekam et al., 2022). Experimental results show that the proposed method consistently and orthogonally improve reasoning performance over the standard distilled models. The improvements come from the reduced hallucinations during long-time thinking, particularly content repetition, which leads to fewer "no answer" phenomena and better overall accuracy. The **main contributions** of this work are:

- Our study reveals the side effect of standard distillation on transferring long CoT reasoning, which results in sub-optimal training of smaller models when using the distilled data (in Section 2.4).
- We propose a novel approach to construct CoT trees from scratch, which not only scales up the solution space but also more closely mimics human-like reasoning patterns. To the best of our knowledge, it is the first attempt of its kind (in Section 2).
- We investigate a set of effective approaches to widen the distillation bottleneck, demonstrating that they are orthogonal and complementary to each other, and robustly applicable to different reasoning tasks and languages (in Section 3&4).

2 Tree-Based CoT Data Construction

We propose a flexible and customizable tree-based CoT data construction method that generates high-quality CoT data from scratch. In this section, we introduce the overall framework in Section 2.1, followed by the thought nodes (Section 2.2), reasoning patterns (Section 2.3), and how to extract CoT data for post-training (Section 2.4).

2.1 Overall Framework

We introduce the tree-based CoT data construction process as shown in Figure 1. This tree structure not only constrains the search space to prevent unbounded expansions but also guides the model to produce reasoning steps (nodes) systematically. For instance, we specify that each node in the search tree corresponds to a particular action role (*e.g.*, *thinking*, *reflection*), and each edge represents a transition to the next step. By constraining the transitions among these nodes, we ensure the search is both tractable and coherent. With the structure in place, we use MCTS to explore the search tree. During each step:

- **Node Selection.** We select a thought node to expand based on MCTS principles, such as upper confidence bound (UCB). If $\mathrm{Child}(n)$ denotes the set of child nodes of node n, then UCB balances exploration and exploitation via a score: $UCB(n_i) = \frac{v(n_i)}{n_{\text{visits}}(n_i)} + C\sqrt{\frac{\ln\left(n_{\text{visits}}(n_{parent})\right)}{n_{\text{visits}}(n_i)}}$, where $v(n_i)$ is an estimated value (or reward) of node n_i, $n_{\text{visits}}(\cdot)$ denotes the visit count, and C is the exploration constant.
- **Expansion.** We expand the selected node by prompting an LLM by adding a thought prompt (detailed in Table 2) that specifies the required action role. The LLM then generates the textual

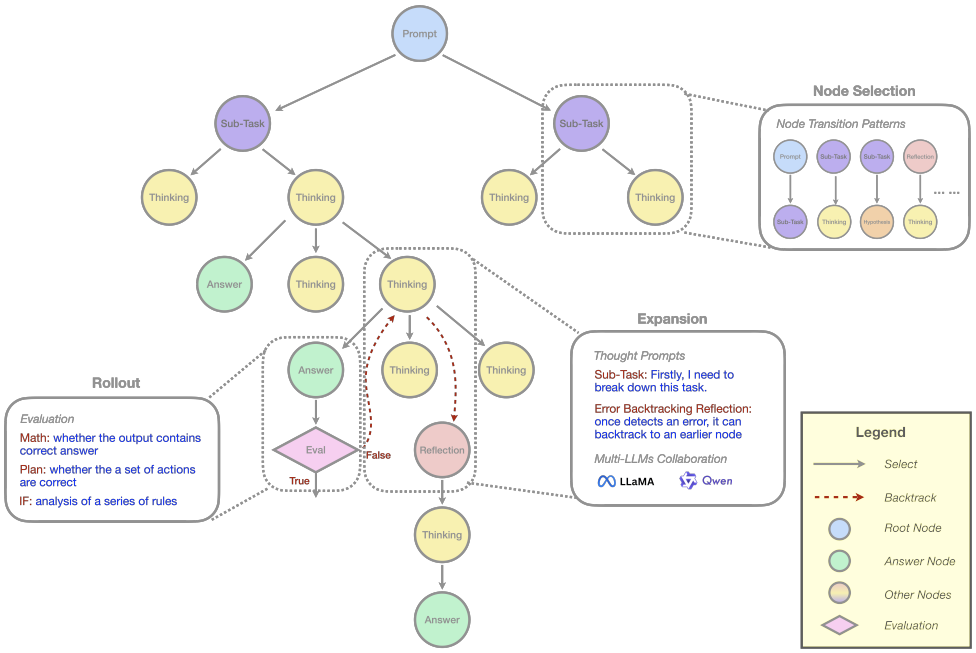

Figure 1: MCTS-based CoT data generation framework. Starting from an initial prompt (root node), the system proceeds through predefined nodes (*e.g.*, *Sub-Task*, *Thinking*, *Reflection*) according to a customizable node transfer matrix. Each node is expanded by prompting either Qwen or Llama, allowing multi-model collaboration. If a wrong answer is detected, we perform error backtracking (pink arrows) to a prior node and trigger *Reflection* in another model, enhancing the overall correctness and diversity of the final reasoning path.

content for that node.

- **Rollout.** If the expansion reaches an *answer* node, we compute a reward based on correctness determined by rules and backpropagate this reward up the tree.

2.2 Thought Node

Definition A Thought Node corresponds to a distinct step or action within the CoT reasoning process. As outlined in Table 2, each node has a dedicated role and prefix prompt that guides the language model to generate specific content or revise previously generated reasoning. This structured design facilitates modular expansion and systematic backtracking within the MCTS framework. Notably, *Thinking* is treated as a special node that does not require any prefix prompt; instead, it admits unconditioned continuation generation to foster open-ended exploration of partial solutions. By combining multiple node types into a coherent tree, we can more effectively elicit and refine multi-step reasoning from the model.

Multi-Model Coordination and Reflection We adopt multi-model coordination to further diversify and correct the generated reasoning paths: 1) For nodes such as *Thinking*, we use Qwen2.5-72B-Instruct to generate logical steps or partial solutions; 2) For *Reflection* nodes, we switch to a different model, e.g., Llama3.1-70B-Instruct, to perform self-checks and corrections.

This separation enhances the reliability of reflection. When the same model that made a mistake also attempts to correct itself, it may fall back on the same erroneous distributional patterns. In our pipeline, if a *Reflection* node detects an error, it can backtrack to a specific earlier node (also configurable in the MCTS design) and request a regeneration of the *Thinking* steps. By alternating

Thought Node	Prompt
Thinking	(continuation generation)
Sub-Task	Firstly, I need to break down this task.
Reflection	Let's check the result. Wait! something is wrong, let's think again.
Hypothesis	I propose the following hypothesis:
Double-Check	Now, I need to check whether all the requirements are met.
Reclarify	To ensure clarity, let me restate the question or issue at hand:
Answer	The answer is:

Table 2: The pre-defined Thought Node. For a selected node, its corresponding prompt is continuously fed to LLMs for MCTS expansion.

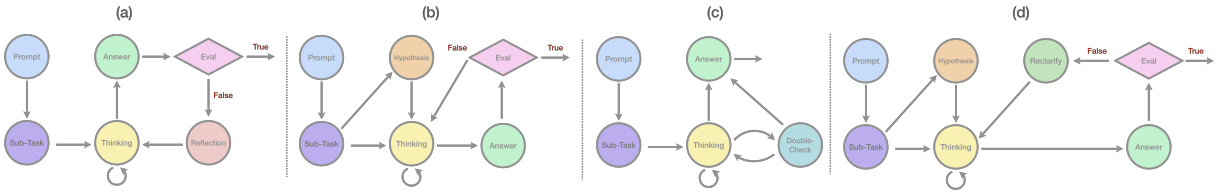

Figure 2: Representative node transition patterns in our search tree. Each sub-figure (**a**–**d**) illustrates a distinct sequence of transitions (*e.g.*, *Sub-Task*, *Thinking*, *Reflection*, *Double Check*, *Hypothesis*) toward arriving at an *Answer* node. These variations allow the search to adaptively expand or backtrack based on correctness checks, thereby generating rich and context-specific chain-of-thought data.

between models, we reduce the risk of repeated mistakes and improve the diversity of exploration.

2.3 Reasoning Pattern

As illustrated in Figure 2, we design a set of *customizable search tree structures* to reflect the diverse ways in which humans reason about different tasks. Each tree is configured to capture a variety of reasoning modes. For instance, in Figure 2(a), we demonstrate a sequence of nodes to solve a question: we first break down the task via a *Sub-Task* node, then perform a general *Thinking* step, and finally provide an *Answer*. We evaluate correctness through rule-based checks: if correct, we output the result; otherwise, we prompt the model to reflect on potential mistakes (entering the *Reflection* node). Here, the model revisits or re-checks its chain of thought, then either formulates new reasoning or proposes a revised answer. This feedback loop repeats until the solution is correct or a preset search limit is reached. Some tasks, however, also require formulating assumptions or provisional conclusions, so in Figure 2(b), we incorporate a *Hypothesis* node immediately after the *Sub-Task* node for tasks that benefit from explicitly positing assumptions or preliminary formulas early on. For example, "Find the sum of all ordered pairs (x, y) of positive integers such that $x + y = 5$." After breaking down the task (*Sub-Task*), the model proposes a hypothesis that x can range from 1 to 4 (with $y = 5 - x$) and enumerates each pair, obtaining a total sum of 20. If the answer is verified correct, the model outputs 20.

In practice, we randomly sample from these different reasoning-flow templates (e.g., Figure 2(a-d)) to ensure we capture diverse, human-like modes of thought. By introducing node transition patterns, our framework produces richer and more flexible CoT data and fosters more robust reasoning in downstream tasks.

Datasets	**Long**	**Middle**	**Short**
GSM8K	5.38%	**5.08%**	**5.08%**
MATH	28.40%	20.60%	**16.20%**
AIME	51.66%	**50.00%**	60.00%
Plan.	6.40%	6.40%	**6.20%**
IF (Zh)	32.30%	4.23%	**1.9%**
IF (En)	22.36%	**3.80%**	4.00%
IF (Ot.)	18.69%	1.70%	**1.59%**

Table 3: Effects of thoughts length(long, medium, and short CoT paths) on model performance across different datasets.

2.4 CoT Data Extraction for Post-Training

Once MCTS completes its exploration, we have a large set of candidate paths. At this point, we must extract final CoT data for SFT or DPO.

- CoT Data for SFT We typically select successful paths that lead to the correct final answer. Depending on the data volume requirements, one can: 1) Pick the highest-reward path according to MCTS; 2) Pick the long or short path that yields the correct answer, if specific chain lengths are desired.
- CoT Data for DPO Constructing DPO data requires both positive and negative examples for each prompt: 1) The positive example is the CoT path that correctly solves the problem, like the SFT data; 2) The negative example is a flawed path (an incorrect final answer) that shares a minimal prefix with the positive path to mitigate excessive overlapping tokens, which can degrade DPO performance.

We find that many existing QA prompts (especially those frequently seen during Qwen or Llama training) are too easy for the models, producing few or no negative paths. As a result, fewer DPO pairs are generated. One can overcome this limitation by using more challenging questions or those that the models have not encountered extensively.

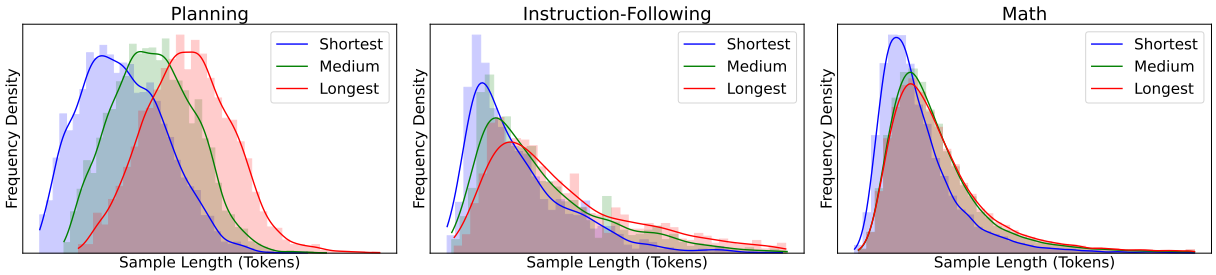

Figure 3: Distribution of data lengths by token count: A comparative analysis of sampling strategies and their correspondence to short, medium, and long data lengths illustrated with histograms and KDE curves

Quantitative Analysis: Formalistic Long-time Thinking We explore the impact of the CoT length distribution on model performance. We experiment with different sampling strategies for DPO datasets, where responses to the same question are ranked and categorized based on their length (Their distributions are shown in Figure 3). As shown in Table 3, the selection of shorter CoT paths leads to a noticeable reduction in ineffective outputs. In addition, we note that shorter reasoning paths tend to mitigate the issue of "formalistic long-time thinking", thus improving the quality of the reasoning output.

3 CoT-Aware Post-Training

Section 2.4 identifies that DPO training is prone to causing the formalistic long-time thinking. In this section, we propose three methods to address this problem: Thoughts Length Balance (Section 3.1), Fine-grained DPO (Section 3.2), Joint Post-training Objective (Section 3.3).

3.1 Thoughts Length Balance

Section 2.4 shows that the length of CoT significantly affects reasoning performance of distilled smaller models at DPO phase. However, in preliminary experiments, we found no such effect on SFT training. Therefore, we propose using longest CoT data during the SFT phase, while using the shortest CoT data during the DPO phase. In practice, we extract all valid paths leading to correct answer nodes from the CoT trees, as there can be multiple correct paths. From these paths, we select examples of varying lengths (long, medium, short) based on token count for SFT. For DPO, the correct paths serve as positive examples, while negative examples are generated by identifying incorrect paths that share the shortest common prefix with the positive examples. This ensures a diverse set of reasoning paths for both SFT and DPO.

Figure 4: The illustration of masking-based DPO, by setting the log probabilities of the common prefix tokens in preference pairs to zero.

3.2 Fine-grained DPO

Recent studies highlight that DPO is sensitive to the length of responses, which can lead to biased reward assessments (Lu et al., 2024; Liu et al., 2024). Longer chosen responses increase the model's tendency to generate longer outputs, while longer rejected responses push the model to move away from such outputs, potentially without reducing their length. These issues are particularly pronounced in long CoT reasoning tasks, where length disparities may undermine DPO's effectiveness in fine-tuning reasoning models.

Conservative DPO Conservative DPO (cDPO) (Mitchell, 2023) adapts the standard DPO framework to handle noisy preference labels, typically encountered when labels may be flipped with a small probability ϵ. The key innovation of cDPO is to modify the target distribution to account for potential label noise, setting the preference probability to $p(y_w \succ y_l) = 1 - \epsilon$. This adjustment reduces the impact of noisy labels by softening the gradient updates, making the model less sensitive to incorrect preferences. Formally, the cDPO loss is defined as:

$$\begin{aligned}\mathcal{L}^{\epsilon}_{\text{DPO}}(\theta, y_w, y_l) = &-(1-\epsilon)\log \hat{p}_\theta(y_w \succ y_l) \\ &-\epsilon \log(1 - \hat{p}_\theta(y_w \succ y_l)),\end{aligned}$$

where $\hat{p}_\theta(y_w \succ y_l)$ is the predicted preference probability. The gradient of this loss combines weighted contributions from both the correct and

Model	Math			Planning	Instruction-Following		
	GSM8K	MATH	AIME	Blocksworld	Zh	En	Other
Llama-3.1-8B-Instruct	85.5	47.0	11.7	10.0	61.5	76.2	67.1
+ Sky-T1	84.8	44.0	6.7	2.0	25.4	31.6	29.7
+ Our Data	**87.4**	**51.4**	**15.0**	**12.4**	**69.2**	**76.6**	**79.1**

Table 4: Comparison of SFT results across various models on multiple different tasks, including math, planning and instruction-following Benchmarks.

incorrect paths, facilitating more stable training under label noise. By upweighting correct preferences and downweighting incorrect ones, cDPO prevents overfitting to noisy data, resulting in more reliable optimization and improved model robustness.

Masking-based DPO To mitigate the adverse effects of shared prefixes inherent in tree-based data, we modify the DPO loss computation by masking out the shared prefix tokens. Specifically, prior to loss calculation, we identify the number of tokens constituting the common prefix between the positive and negative samples and adjust the loss mask by setting the corresponding entries for these shared tokens to zero—analogous to the treatment of padding tokens in standard loss formulations, as shown in Figure 4. This ensures that the shared prefix tokens do not contribute to the gradient computation, allowing the model to focus on the differentiating segments of the outputs and better distinguish between valid and invalid reasoning paths. This strategy provides a fine-grained adjustment to the DPO objective, enhancing optimization in settings with substantial prefix overlap.

3.3 Joint Post-training Objective

In model training, the typical approach follows a sequential training paradigm, first conducting SFT followed by RLHF or DPO. However, this sequential training process is suboptimal due to the inherent trade-off between SFT and RLHF/DPO, where the model tends to forget the content learned in the first stage as it progresses to the second. Even regularization methods like KL divergence cannot fully mitigate the forgetting caused by the distribution shift from the SFT dataset to the preference-based dataset, as highlighted by (Fernando et al., 2025). A similar phenomenon is observed in our work, where such forgetting contributes to the emergence of formalistic long-time thinking in distilled models. To address this, we introduce SFT loss during the DPO training phase to alleviate the performance degradation resulting from the switch in training methodologies. The final loss function is thus modified as: $\mathcal{L} = \mathcal{L}_{\text{DPO}} + \alpha\,\mathcal{L}_{\text{SFT}}$, where the hyperparameter α enables a better trade-off between SFT and preference learning, helping to maintain consistency in the model's reasoning patterns throughout the training process. This adjustment ensures more robust and stable performance across stages.

4 Experiments

4.1 Experimental Setup

Models We start with the baseline model, "Our LRM (SFT)," which is Llama-3.1-8B fine-tuned on our CoT data. Direct Preference Optimization (DPO) is applied next, followed by Data Length Balance. Conservative DPO (cDPO) is then added, and a Joint Loss function combining DPO and Supervised Fine-Tuning (SFT) loss is incorporated. Finally, masking-based DPO is applied. Each of these methods is sequentially added to the baseline, as shown in Table 5.

Benchmark We evaluate our approach on five benchmarks, each capturing different reasoning challenges. AIME focuses on higher-level math with 60 questions from 2023 and 2024, while GSM8K (Cobbe et al., 2021a) features elementary-to-intermediate arithmetic tasks. MATH500 (Lightman et al., 2023) presents a wide range of advanced mathematical problems, testing deeper analytical thinking. For sequential decision making, we adopt the classical Blocksworld (Valmeekam et al., 2022) planning domain from the International Planning Competitions (IPC). Lastly, Multi-IF (He et al., 2024) assesses multi-turn instruction following in eight languages, encompassing 4,501 multilingual, three-turn conversations.

4.2 Experimental Results

Constructed Data Validation We apply our constructed CoT data to smaller models such as Llama-8B, comparing it against the Sky-T1 8B model, which utilizes a distillation pipeline based on QwQ

Model	Math			Planning	Instruction-Following		
	GSM8K	MATH	AIME	Blocksworld	Zh	En	Other
Baseline							
Our LRM (SFT)	87.4 (*0.23%*)	**51.4** (**5.40%**)	**15.0** (**30.00%**)	12.4 (**1.80%**)	69.2 (**0.77%**)	76.6 (**1.69%**)	**79.1** (**1.08%**)
+ DPO	86.2 (*6.37%*)	41.8 (*31.80%*)	8.3 (*55.00%*)	2.0 (*93.60%*)	5.7 (*91.54%*)	6.3 (*90.93%*)	6.7 (*92.22%*)
Our Methods							
+ Data Balance	86.8 (*5.08%*)	28.0 (*46.40%*)	6.6 (*65.00%*)	6.8 (*44.60%*)	43.4 (*30.77%*)	44.7 (*44.73%*)	42.4 (*45.28%*)
+ cDPO	**87.5** (*3.71%*)	48.6 (*15.00%*)	**15.0** (*45.00%*)	4.4 (*47.40%*)	61.9 (*11.15%*)	66.4 (*15.61%*)	67.7 (*15.40%*)
+ Joint Loss	86.8 (*0.38%*)	48.6 (*8.60%*)	10.0 (*31.67%*)	8.6 (*9.00%*)	**72.3** (*1.15%*)	**78.9** (*1.90%*)	78.1 (*2.22%*)
+ Masking	87.2 (**0.15%**)	51.0 (*5.80%*)	8.0 (*38.33%*)	**12.6** (*10.20%*)	72.0 (*1.15%*)	77.2 (*1.90%*)	**79.1** (*1.36%*)

Table 5: Performance comparison among different methods. The best performance is boldfaced, while the second best is underlined. The numbers in parentheses indicates the ratio of instances where no answer is obtained in the specified format.

and proves effective on larger models (32B). While Sky-T1 demonstrates competitive performance on large models, it faces challenges when scaled down to 8B models due to the inherent limitations in context processing and reasoning capabilities. In contrast, our CoT data, specifically designed to address these limitations, leads to substantial improvements in smaller models, particularly in tasks involving arithmetic reasoning such as GSM8K and MATH, as well as more complex open-ended tasks like AIME and Blocksworld, as shown in Table 4. This validates the effectiveness of our constructed data in advancing the performance of small models across a wide range of reasoning tasks.

Main Results As shown in Figure 5, we progressively adding various techniques described in Section 3 to address the challenges identified during DPO. Initially, we observe that DPO causes a significant increase in output length, which results in a marked drop in model performance due to the high proportion of samples without answers. To mitigate this issue, we explore several strategies, including the data balance, applying cDPO to reduce the impact of noisy labels, SFT Loss for multi-objective training to prevent catastrophic forgetting, and masking shared prefixes during loss calculation to reduce overemphasis on redundant tokens. Our results show that these adjustments achieves a notable improvement in reasoning tasks, particularly in planning and instruction-following, while maintaining competitive performance on mathematical benchmarks. The performance improvements can be primarily attributed to the reduction of formalistic long-time thinking, which is a major source of inefficiency in reasoning. By addressing this issue, our model exhibits a stronger ability to generate meaningful and correct answers instead of producing excessive, irrelevant reasoning steps. This leads to a substantial enhancement in overall model effectiveness, with improvements of coherent reasoning and accurate outputs. The integration of these methods ensures that our model achieves robust and efficient performance across a wide range of reasoning tasks, Contributing to the application of DPO technology in LRMs.

4.3 Effects of Joint DPO Loss

In this study, we explore the effects of combining DPO loss with SFT loss within a joint post-training objective, as outlined in Section 3.4. Our goal is to stabilize the training process and mitigate issues like over-optimization observed in pure DPO. To this end, we experiment with varying values of the hyperparameter alpha, which controls the weight balance between the DPO and SFT losses. As shown in Table 6, our results indicate that alpha=1 provides the best trade-off, as smaller values still lead to some degree of catastrophic forgetting, while larger values reduce the effectiveness of preference alignment, thereby diminishing the efficiency of the valuable preference dataset. Conse-

Datasets	CDPO	+0.5	+1.0	+1.5	+2.0
GSM8K	**87.5** (*3.71%*)	86.5 (*0.53%*)	86.8 (*0.38%*)	85.5 (**0.08%**)	85.6 (**0.08%**)
MATH	48.6 (*15.00%*)	**50.0** (*10.20%*)	48.6 (*8.60%*)	48.4 (**0.08%**)	48.0 (*9.00%*)
AIME	**15.0** (*45.00%*)	11.6 (*35.00%*)	10.0 (**31.67%**)	6.6 (*36.67%*)	11.6 (**31.67%**)
Plan.	4.4 (*47.40%*)	7.8 (*12.80%*)	**8.6** (*9.00%*)	7.6 (**6.40%**)	8.4 (*7.40%*)
IF (Zh)	61.9 (*11.15%*)	68.8 (*2.31%*)	**72.3** (*1.15%*)	68.4 (*3.46%*)	70.7 (**0.77%**)
IF (En)	66.4 (*15.61%*)	76.1 (*4.22%*)	**78.9** (**1.90%**)	77.2 (*2.74%*)	78.2 (*2.32%*)
IF (Ot.)	67.7 (*15.40%*)	78.0 (*1.99%*)	78.1 (*2.22%*)	**79.2** (**1.82%**)	78.9 (*1.93%*)

Table 6: Effects of joint loss (combining DPO loss and SFT loss with a weight factor α in $\mathcal{L} = \mathcal{L}_{\text{DPO}} + \alpha\mathcal{L}_{\text{SFT}}$) on model performance across different datasets with varying hyperparameter settings.

quently, we adopt the combined $\mathcal{L} = \mathcal{L}_{\text{DPO}} + \mathcal{L}_{\text{SFT}}$ as the configuration for subsequent experiments.

4.4 MCTS Inference Exploration

As an additional experiment, we explored the impact of applying MCTS at the inference stage. As shown in Table 7, we use Test@N to denote the percentage of problems solved correctly at least once when allowing the model to make N separate guesses for each problem.(Cobbe et al., 2021b) We evaluated solve rates at Test@1, Test@8, and Test@32 on the MATH dataset. Specifically, at Test@8 and Test@32, the MCTS-based approach outperforms the model without MCTS inference, demonstrating its ability to expand the solution space and leverage test-time scaling effectively.

5 Related Work

Reasoning Models Several prior works have explored various approaches, including CoT fine-tuning (Wei et al., 2022), reinforcement learning (RL) (Kumar et al., 2024), More recently, Large reasoning models (LRMs) such as OpenAI o1 (OpenAI, 2024), and DeepSeek-R1 (Guo et al., 2025) have shown remarkable reasoning capabilities by scaling test-time compute and generating long CoT.

Knowledge Distillation Knowledge distillation has also been crucial for transferring reasoning capabilities from large models to smaller ones. Guo et al. (2025) investigated direct distillation, where smaller models are fine-tuned on CoT data generated by large reasoning models, yielding significant improvements on math benchmarks. However, distillation faces challenges such as overfitting and the inheritance of biases from large models, which can lead to suboptimal reasoning patterns in smaller models. Research by Chen et al. (2024) and Wang et al. (2025) highlighted that long CoT reasoning could introduce learning difficulties and biases, impacting the performance of smaller models.

Model	Test@1	Test@8	Test@32
Llama-3.1-8B-Instruct	47.0	67.6	**75.8**
Our Best Model	51.0	70.2	**79.2**
+ MCTS Decode	51.0	70.8	**82.8**

Table 7: Performance on MATH Dataset: Test@1, Test@8, and Test@32 Results. Test@N denotes the percentage of problems solved correctly at least once when the model is allowed to make N separate guesses for each problem.

Monte Carlo Tree Search MCTS has been proposed as a promising solution to improve reasoning quality. (Qi et al., 2024) and (Tian et al., 2024) applied MCTS for reasoning in LLMs.They use MCTS to guide the model to generate outputs with long Chains of Thought (CoT), and then use these outputs for continued training.

6 Conclusion

We explore strategies for transferring long CoT reasoning to smaller models, addressing learning difficulties and bias inheritance common in standard distillation. We propose a MCTS framework that generates tree-based reasoning paths from scratch, enabling flexible data generation and reducing reliance on large teacher models. We enhance performance through CoT-aware post-training, combining supervised fine-tuning, direct preference optimization, and masking-based strategies. Experiments on diverse benchmarks—covering exam tasks, multilingual instruction following, and real-world planning—show that MCTS-generated data and targeted optimization reduce formalistic overthinking and content repetition. By balancing reasoning length, introducing multi-model coordination for reflection, and using a joint loss approach for preference optimization, our model demonstrates robust performance across various reasoning tasks. These findings highlight the importance of well-designed data and post-training strategies in improving the efficiency and reliability of smaller-scale reasoning models.

Limitations

Our MCTS-based method for constructing CoT data moves beyond straightforward distillation pipelines by explicitly building a search tree, guiding the LLM to generate multi-step, human-like reasoning. Despite its advantages, the method can be computationally expensive as task complexity increases. Future optimizations could include: 1) Pruning Strategies: Prune branches of the tree that appear suboptimal based on a partially learned reward or confidence metric. 2) More Diverse Model Rotations: Beyond reflection vs. thinking, *multiple* specialized models (or specialized parameter shards within a large model) could be employed for different reasoning skills (e.g., logic vs. creative).

Ethics Statement

We take ethical considerations very seriously, and strictly adhere to the ACL Ethics Policy. All datasets used in this paper are publicly available. We ensure that the findings and conclusions of this paper are reported accurately and objectively.

References

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Heshan Fernando, Han Shen, Parikshit Ram, Yi Zhou, Horst Samulowitz, Nathalie Baracaldo, and Tianyi Chen. 2025. Mitigating forgetting in llm supervised fine-tuning and preference learning. *Preprint*, arXiv:2410.15483.

Heshan Devaka Fernando, Han Shen, Parikshit Ram, Yi Zhou, Horst Samulowitz, Nathalie Baracaldo, and Tianyi Chen. 2024. Mitigating forgetting in LLM supervised fine-tuning and preference learning. *CoRR*, abs/2410.15483.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. 2024. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*.

Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. O1 replication journey–part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *arXiv preprint arXiv:2411.16489*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Wei Liu, Yang Bai, Chengcheng Han, Rongxiang Weng, Jun Xu, Xuezhi Cao, Jingang Wang, and Xunliang Cai. 2024. Length desensitization in direct preference optimization. *Preprint*, arXiv:2409.06411.

Junru Lu, Jiazheng Li, Siyu An, Meng Zhao, Yulan He, Di Yin, and Xing Sun. 2024. Eliminating biased length reliance of direct preference optimization via down-sampled kl divergence. *Preprint*, arXiv:2406.10957.

Eric Mitchell. 2023. A note on dpo with noisy preferences & relationship to ipo.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. [Accessed 19-09-2024].

Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *Preprint*, arXiv:2408.06195.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, et al. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing. *Preprint*, arXiv:2404.12253.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Neural Information Processing Systems*.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Zhichao Wang, Bin Bi, Zixu Zhu, Xiang-Bo Mao, Jun Wang, and Shiyu Wang. 2024. UFT: unifying fine-tuning of SFT and RLHF/DPO/UNA through a generalized implicit reward function. *CoRR*, abs/2410.21438.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405*.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, et al. 2024. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*.