

Thinking Preference Optimization

Anonymous ACL submission

Abstract

Supervised Fine-Tuning (SFT) has been a go-to and effective method for enhancing long chain-of-thought (CoT) reasoning in relatively small LLMs by fine-tuning them with long CoT responses from larger LLMs¹. To continually improve reasoning abilities, we can either collect new high-quality long CoT reasoning SFT data or repeatedly train on existing SFT datasets. However, acquiring new long CoT SFT data is costly and limited, while repeated training often results in a performance plateau or decline. To further boost the performance with the SFT data, we propose Thinking Preference Optimization (ThinkPO), a simple yet effective post-SFT method that enhances long CoT reasoning without requiring new long CoT responses. Instead, ThinkPO utilizes readily available or easily obtainable short CoT reasoning responses as rejected answers and long CoT responses as chosen answers for the same question. It then applies direct preference optimization to encourage the model to favor longer reasoning outputs. Experiments show that ThinkPO further improves the reasoning performance of SFT-ed models, e.g. it increases math reasoning accuracy of SFT-ed models by 8.6% and output length by 25.9%. Notably, ThinkPO is capable of continually boosting the performance of the publicly distilled SFT model, e.g., increasing the official DeepSeek-R1-Distill-Qwen-7B’s performance on MATH500 from 87.4% to 91.2%. Our code is available at <https://anonymous.4open.science/r/ThinkPO-757B/>.

1 Introduction

The reasoning capability of LLMs is crucial for their applicability in complex problem-solving tasks. Improving the reasoning abil-

¹Deepseek official distilled models DeepSeek-R1-Distill, OpenThinker-7B, Sky-T1-32B, and Bespoke-Stratos-7B was trained in this way.

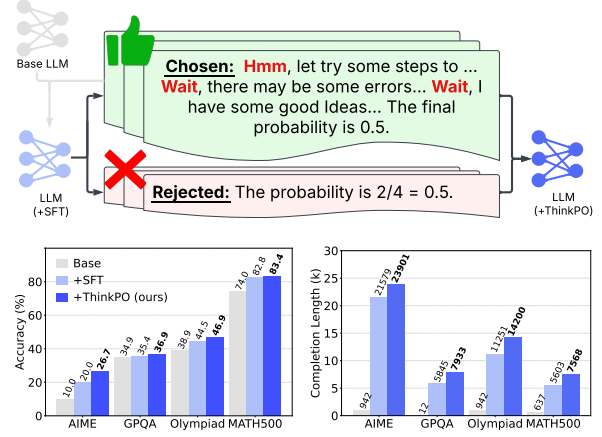


Figure 1: The illustration of our method ThinkPO and its performance on math reasoning tasks. **Top:** Our ThinkPO enhances fine-tuned LLMs (+SFT) by promoting detailed problem-solving—using long chain-of-thought reasoning answers as positive (chosen) samples and short chain-of-thought reasoning answers as negative (rejected) samples. **Bottom Left:** ThinkPO significantly boosts performance across mathematical benchmarks (e.g., 83.4% on MATH500 vs. 82.8% for +SFT and 74.0% for the Base model). **Bottom Right:** ThinkPO generates more detailed solutions, with average completion lengths on AIME increasing from 0.94K to 21.57K to 23.9K tokens. These results underscore Think Preference Optimization’s effectiveness in fostering and enhancing advanced mathematical reasoning.

ity of large language models is one of the current research hotspots. Many approaches have emerged in the open-source community that enhance relatively small models’ reasoning ability through SFT. For example, Sky-Thought (Schulman et al., 2017), Bespoke-Stratos (Labs, 2025) and OpenThinker-7B (Team, 2025b) have built long reasoning datasets to fine-tune models fully, aiming to improve model reasoning capabilities. Further advancements can be seen in models like s1 (Muenighoff et al., 2025) and LIMO (Ye et al., 2025), which focus on the sophisticated design of long reasoning datasets to enhance reasoning capabilities.

Despite the success of supervised fine-tuning,

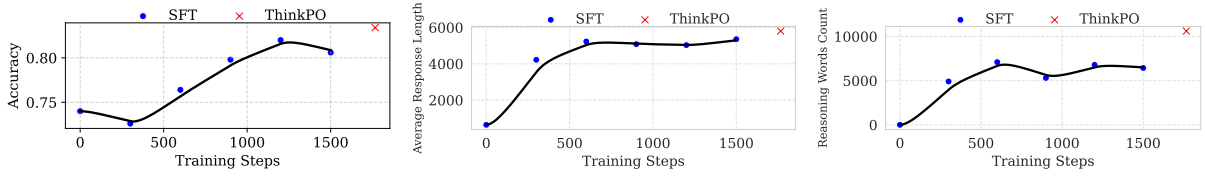


Figure 2: Analysis of accuracy(Left), average response length(Middle) and reasoning-supportive words count(Right, like wait, hmm, etc) in SFT and ThinkPO process. We evaluate the model on MATH500 every 300 steps and record all the three metrics. In the early training stages, all of them improve significantly. However, in the later stages (e.g., after 1200 steps), the model’s performance gradually plateau. When ThinkPO is applied, we see additional improvements in all of the three aspects, demonstrating the effectiveness of Thinking Preference Optimization.

continually improving the reasoning abilities of the STF-ed model faces the following challenges: (1) **high resources cost needed to collect new long reasoning response**: Training a stronger reasoning model first requires collecting new large-scale, diverse, and meticulously designed long-reasoning questions. Then, responses to these long reasoning problems need to be collected from large-scale models, such as DeepSeek-R1. However, collecting questions and responses requires significant computational power and human resources, making the process both expensive and labor-intensive, especially in domains such as healthcare or low-resource languages, where gathering these input questions is non-trivial and models like DeepSeek R1 may also struggle to produce accurate responses. Furthermore, (2) **repeatedly fine-tuning LLMs on existing long responses face Performance bottleneck**: As a compromise, one might repeatedly train on a limited long reasoning dataset, but this approach typically leads to a performance plateau or even decline. In Figure 2, we observe that when training with a fixed amount of long-reasoning data for multiple epochs, model’s average output length and accuracy increase significantly in the early stages but slow down or even plateau in later stages. According to the test-time scaling principle (Snell et al., 2024; Welleck et al., 2024), increasing the compute at test time generally enhances reasoning ability. However, the limited long-reasoning dataset is insufficient to further improve LLMs’ reasoning capability in later stages of SFT.

To overcome the performance bottleneck and better utilize existing long reasoning data, we propose **Thinking Preference Optimization**: a simple yet efficient method to further enhance model reasoning ability after supervised fine-tuning (SFT). Our approach utilizes short CoT reasoning responses—which are already available or easy to acquire—as rejected answers and *existing* long CoT responses

as chosen answers for the same question, and employs Direct Preference Optimization to train models. This encourages models to prefer longer and more structured reasoning processes, thereby improving reasoning abilities *without acquiring additional high-quality long CoT responses*.

Figure 1 presents the framework of ThinkPO along with the experimental results. We first fine-tune a Qwen base model using the long CoT data to obtain an SFT-ed model (+SFT), and then we further train it using ThinkPO (+ThinkPO). The results in Figure 1 clearly show that our method improves mathematical reasoning ability across four datasets. Additionally, our method increases the average response length on all four datasets, aligning with the test-time scaling trend. For example, ThinkPO increases the math reasoning accuracy of SFT-ed models by 8.6% and the output length by 25.9%. Notably, ThinkPO increases the official DeepSeek-R1-Distill-Qwen-7B’s performance on MATH500 from 87.4% to 91.2%. The main contributions are summarized as follows:

- We propose Thinking Preference Optimization (ThinkPO) to maximize the value of existing long reasoning data, which successfully further enhances SFT-ed LLMs’ reasoning performance without additional long CoT responses.
- Our method continuously improves the performance of public R1-distilled models, including the DeepSeek-R1 official distilled models.
- We release our dataset, codes, and model weights to facilitate further research.

2 Thinking Preference Optimization

2.1 Motivations

This section introduces the motivations behind Thinking Preference Optimization. SFT with fixed long-reasoning datasets is an effective method for enhancing a model’s reasoning ability. However,

further improvement of the model’s reasoning ability during the later stages faces a bottleneck. In such cases, by using short reasoning data as rejected samples and long reasoning texts from SFT as chosen samples for DPO training, it is possible to further leverage the high-quality SFT reasoning data to boost the model’s reasoning performance with minimal additional data resources.

First, we finetune Qwen-2.5-7B-Instruct model using Bespoke-Strato-dataset (Labs, 2025), which includes 17k long reasoning data distilled from Deepseek-R1. During training, we track the model’s average output length, accuracy and reasoning-supportive words count (like wait, hmm) at different steps on the Math500 dataset. These are visualized by fitting curves. When calculating the model’s average output length, we only considered valid sentences, excluding duplicates or sentences with formatting errors. The results on other datasets could be found in Appendix A.2.

In Figure 2, in the early stages of SFT, the model’s average output length, accuracy and reasoning-supportive words count show significant improvements. This aligns with the test-time scaling phenomenon (Snell et al., 2024; Welleck et al., 2024), where a model’s reasoning ability generally improves as its output length increases. Many approaches enhance reasoning ability by fine-tuning models to generate longer responses. However, in the later stages of SFT, average response length, accuracy and reasoning-supportive words count plateau, indicating a performance bottleneck.

To further enhance the model’s reasoning ability, we can apply DPO, which encourages the model to favor longer outputs. By treating long-reasoning responses as chosen samples and short-reasoning responses as rejected samples, this approach improves the model’s reasoning ability without significantly increasing long-reasoning dataset size, thereby boosting its reasoning performance.

2.2 Training Pipeline

The training process in Thinking Preference Optimization consists of two stages: Reasoning SFT (Supervised Fine-Tuning) stage and Reasoning DPO (Direct Preference Optimization) stage.

In the Reasoning SFT stage, long-reasoning responses are collected for each question to construct the dataset $\mathcal{D}_{sft}$. The base model is then fine-tuned on $\mathcal{D}_{sft}$ to acquire advanced reasoning capabilities, which helps to prepare the model for next stage.

In the second stage, the model is further en-

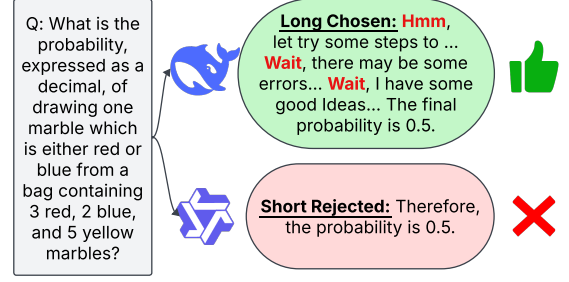


Figure 3: Data Collection Process: we use Deepseek R1 to generate long reasoning answers as chosen samples and Qwen 2.5-7B-Math to generate short reasoning answers as rejected samples, collecting datasets for DPO Training. Compare with short reasoning data, long reasoning answers includes many reasoning-supportive discourse markers, such as wait, hmm, and other hesitation cues, which can improve the model’s reasoning ability.

couraged to generate extended reasoning using the Direct Preference Optimization (DPO) (Rafailov et al., 2024) approach. First, the long-reasoning data from the initial stage is used as the chosen responses. Then, a smaller model with normal Reasoning ability, such as Qwen-2.5-7B-Math (Yang et al., 2024b), is utilized to generate shorter reasoning responses as rejected samples. To ensure data quality, both long and short reasoning responses undergo filtering, including correctness validation. This process results in the dataset $\mathcal{D}_{dpo}$. Finally, the model trained in the first stage is fine-tuned on $\mathcal{D}_{dpo}$ using DPO, encouraging the model to generate longer outputs while enhancing its reasoning ability. Training pipeline is visualized as Figure 1.

2.3 Data Curation

The dataset $\mathcal{D}_{sft} = \{(q, o_{long})\}_N$ is based on bespoke stratos dataset (Labs, 2025). They used DeepSeek-R1 as the teacher reasoning model instead of QwQ-32B-Preview to generate long reasoning response o_{long} and employed GPT-4o-mini in place of Sky-thought T1’s (Team, 2025a) parsing logic to filter out incorrect mathematical solutions.

For the dataset $\mathcal{D}_{dpo} = \{(q, o_{long}, o_{short})\}_N$ in the second stage, we first use Qwen2.5-Math-7B-Instruct (Yang et al., 2024b) to generate a short reasoning response o_{short} , pairing it with the long reasoning response o_{long} in $\mathcal{D}_{sft}$. Then we retain most of the samples where Qwen2.5-Math-7B-Instruct’s answer matched DeepSeek R1’s answer. The more details are in Appendix A.3. The dataset is collected through a straight foreword and simple process of gathering short-reasoning data, which did

Table 1: Accuracy and Average Response Length comparison for Our finetuned Qwen-2.5-7B-Instruct before and after ThinkPO. The "Improv." column shows the percentage change of **Ours** over the model. After applying ThinkPO, its accuracy and length almost improve across datasets, further validating the effectiveness of ThinkPO.

Dataset	Accuracy				Average Response Length			
	Base	+SFT	+ThinkPO	Improv.(%)	Base	+SFT	+ThinkPO	Improv.(%)
MATH500	74.0	82.8	83.4	0.7%	637	5603	7568	35.0%
AIME	10.0	20.0	26.7	33.5%	942	21579	23901	10.7%
GPQA	34.9	35.4	36.9	4.2%	12	5845	7933	35.6%
GSM8K	90.1	93.9	93.0	-0.9%	260	1310	1599	22.1%
Olympiad	38.9	44.5	46.9	5.4%	942	11251	14200	26.2%
Avg.	49.6	55.3	57.4	8.6%	558	9117	11040	25.9%

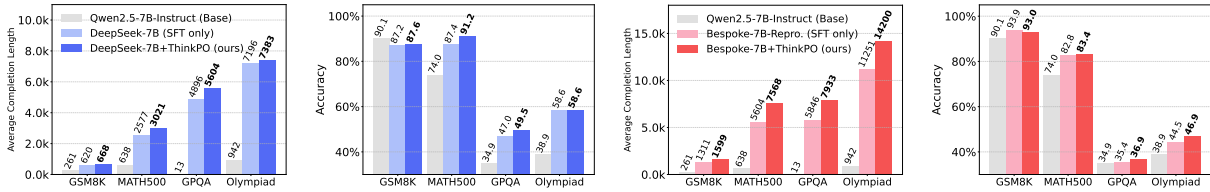


Figure 4: Visualization of improvements on Accuracy and Average Response Length of DeepSeek-R1-Distill-Qwen-7B (Left) and our finetuned Qwen2.5-7B-Instruct (Right) on four datasets After ThinkPO. ThinkPO could improve DeepSeek-7B’s and our finetuned Qwen2.5-7B’s accuracy and output lengths almost across all the datasets

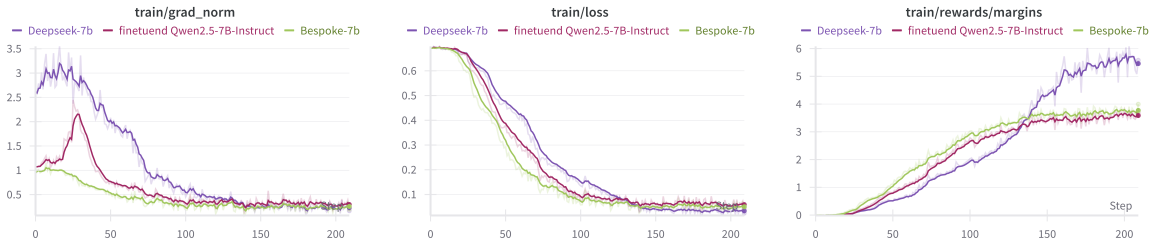


Figure 5: Training loss, gradient norm, and margin curves for DeepSeek-R1-Distill-Qwen-7B, Bespoke-Stratos-7B and our finetuned Qwen2.5-7B-Instruct during Thinking Preference Optimization phase.

not require significant resources, compared to collecting high-quality long-reasoning data.

Since we select responses generated by Qwen2.5-Math-7B-Instruct based solely on final answer matching, this approach may inadvertently include subtle incorrect generations that are not easily detected. To address this, we utilize the OpenR1-Math-220k dataset, which contains DeepSeek-R1 responses collected specifically for questions in NuminaMath-CoT dataset. This ensures that both the *short* and *long* reasoning samples in the dataset are verified to be correct. We conduct the same set of experiments on this dataset and observe consistent results, as shown in Appendix A.6.

3 Experiments

3.1 Experimental Setup

To evaluate model’s reasoning ability, we select five different test sets: MATH500 (Lightman

et al., 2023), AIME2024², GPQA-Diamond (Rein et al., 2023), GSM8K (Cobbe et al., 2021), and Olympiad Bench Math (He et al., 2024). These test sets primarily consist of mathematical reasoning problems, with GPQA-Diamond also including problems from physics, chemistry, and biology. The difficulty levels of these test sets vary significantly, with GSM8K being the easiest while AIME2024 is the most challenging. This diverse selection ensures a comprehensive assessment of the model’s reasoning capability across different levels of difficulty, from fundamental arithmetic to complex problem-solving with different difficulty.

When generating responses, we set the temperature as 0.7. For results on other temperatures, please refer to Appendix A.1. We present our chosen hyper-parameters of ThinkPO in Appendix A.4.

²AIME2024 is a math competition for high school students, acting as a qualifier for the USAMO.

Table 2: Accuracy and Average Response Length comparison for Deepseek-7B and Bespoke-7B before and after ThinkPO. Qwen2.5-7B-Instruct shows the base performance, Deepseek-7B/Bespoke-7B report performance after SFT, and the "Improv." column shows the percentage change of **Ours** over Deepseek-7B/Bespoke-7B. AIME22-24 refers to the AIME datasets from 2022 to 2024, which contains more data samples to evaluate models.

DeepSeek-R1-Distill-Qwen-7B (Deepseek)						
Dataset	Accuracy			Average Response Length		
	Deepseek (SFT)	Ours (+ThinkPO)	Improv. (%)	Deepseek (SFT)	Ours (+ThinkPO)	Improv. (%)
MATH500	87.4	91.2	4.3%	2577	3021	17.2%
AIME22-24	43.3	46.4	7.2%	12824	13570	5.8%
GPQA	47.0	49.5	5.3%	4895	5604	14.5%
GSM8K	87.2	87.6	0.5%	619	668	7.9%
Olympiad	58.6	58.6	0.0%	7196	7383	2.6%

Bespoke-Stratos-7B (Bespoke)						
Dataset	Accuracy			Average Response Length		
	Bespoke (SFT)	Ours (+ThinkPO)	Improv. (%)	Bespoke (SFT)	Ours (+ThinkPO)	Improv. (%)
MATH500	84.0	82.8	-1.4%	5696	6404	12.4%
AIME22-24	16.0	17.6	10.0%	17819	18901	6.1%
GPQA	37.9	43.4	14.5%	5968	7301	22.3%
GSM8K	92.9	93.3	0.4%	1404	1755	25.0%
Olympiad	44.1	48.5	10.0%	11140	12204	9.6%

3.2 Effectiveness of ThinkPO

This experiment primarily analyzes the average response length, accuracy and reasoning-supportive words count during both SFT and DPO processes to validate the effectiveness of Thinking Preference Optimization (ThinkPO). By tracking these metrics, we aim to demonstrate how ThinkPO enhances the model’s reasoning ability by encouraging longer, more structured outputs, ultimately leading to improved reasoning performances.

First, we fine-tune Qwen-2.5-7B-Instruct with Bespoke-Stratos-Dataset. Subsequently, we apply ThinkPO to enhance the model’s reasoning ability. The final results are shown in Table 1. Our fine-tuned model achieves scores across the five datasets that are almost identical to Bespoke-Stratos-7B, which is also finetuned on Bespoke-Stratos-Dataset, confirming the correctness of our SFT process. Furthermore, after applying ThinkPO, our model demonstrates improvements on almost all the datasets, validating the effectiveness of ThinkPO in enhancing and improving LLM reasoning ability.

Additionally, we analyze average response length and reasoning-supportive words (like *wait*, *hmm*, etc) at different steps during both SFT and ThinkPO. We record the model’s average response length, accuracy and reasoning-supportive words

(like *wait*, *hmm*, etc) count on Math500 at different training steps, distinguishing between the SFT and ThinkPO. When calculating average response lengths, we exclude duplicate or incomplete responses to ensure accuracy. Additionally, when counting reasoning-supportive words, we only consider correct answers to prevent excessive occurrences of filler words like “wait” due to underthinking (Chen et al., 2024; Kirk et al., 2023; Wang et al., 2025). The results are visualized in Figure 2.

At the initial stage of SFT, the model’s reasoning ability improves significantly. In the later stages (like after 1200 steps), three metrics gradually plateau, indicating that the model may have reached a local optimum. However, after applying ThinkPO, average response length, reasoning-supportive words count and accuracy improve, showing the effectiveness of ThinkPO in overcoming this stagnation. We visualize the trend of three metrics on other datasets in Appendix A.2.

3.3 ThinkPO Continually Improves Reasoning Ability of Public Distilled Models

We select two open-source reasoning models and perform ThinkPO training using $\mathcal{D}_{dpo}$. Specifically, we chose DeepSeek-R1-Distill-Qwen-7B and

Table 3: Results of Models with Different Sizes (3B, 7B, 14B) on the Qwen-2.5 Family. We evaluate models of different sizes (3B, 7B, 14B) trained with Supervised Fine-Tuning (SFT) and Think Preference Optimization (ThinkPO). Models are fine-tuned on the Bespoke-Strato-Dataset for 1 epoch. As model size increases, accuracy improves across all five test datasets. After ThinkPO training, accuracy improves consistently for models of all sizes, including the smallest (3B), demonstrating that ThinkPO enhances reasoning ability across different model scales.

	Qwen 2.5-3B			Qwen 2.5-7B			Qwen 2.5-14B		
	+SFT	+ThinkPO	Improv.	+SFT	+ThinkPO	Improv.	+SFT	+ThinkPO	Improv.
MATH500	53.6	54.6	1.8%	73.0	74.6	2.2%	83.2	85.6	2.9%
AIME	3.30	6.7	100%	16.7	13.3	-20.3%*	23.3	33.3	42.9%
GPQA	26.3	27.3	3.8%	32.3	36.4	12.7%	45.5	44.0	-3.2%
GSM8K	80.4	81.1	0.8%	88.2	88.9	0.9%	93.7	93.9	0.2%
Olympiad	20.0	22.0	10.0%	35.3	37.2	5.3%	49.9	52.1	4.4%

* Since AIME2024 contains only 30 questions, even a small difference in the number of correct answers can lead to significant fluctuations in accuracy, making the decline appear larger than it actually is.

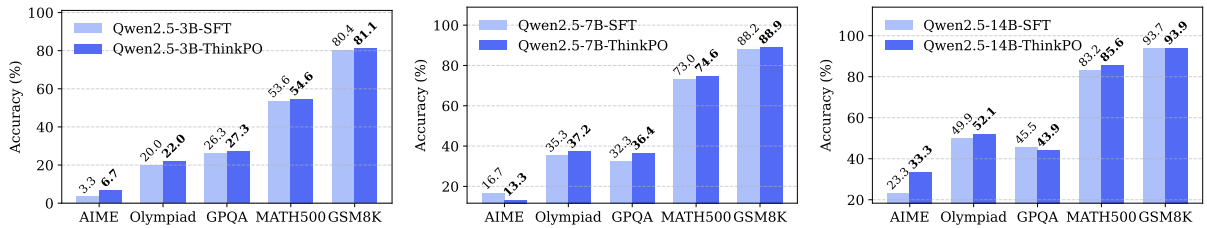


Figure 6: Visualization of improvements on Accuracy and Average Response Length of models in the same family series from different sizes (Qwen-2.5-3B, Qwen-2.5-7B and Qwen-2.5-14B) on five datasets after ThinkPO. ThinkPO could improve models’ accuracy and output lengths almost across all the datasets, regardless of sizes

Table 4: Results of ThinkPO on the model finetuned with a short-Reasoning Dataset. We select a short-chain reasoning dataset of the same size as the Bespoke-Stratos dataset and fine-tune Qwen-2.5-7B for 3 epochs. Models trained with reasoning-style datasets, regardless of response length, can benefit from ThinkPO to enhance and improve their reasoning capability

	Short +SFT	Our +ThinkPO	Improv. %
MATH500	57.8	59.0	2.4%
AIME	0.0	3.3	100%
GPQA	30.3	31.3	3.3%
GSM8K	83.4	85.1	2.0%
Olympiad	23.3	23.6	1.2%

for a slight decline on the MATH500 dataset. Notably, the improvements on Olympiad Bench Math and GPQA-Diamond reach around 5%. DeepSeek-R1-Distill-Qwen-7B, with the exception of a decline on AIME2024, shows consistent or slightly improved accuracy. Specifically, on MATH500, the accuracy improves from 87.4% to 91.2%.

In addition to accuracy, average response length of DeepSeek-R1-Distill-Qwen-7B is increased by around 500 tokens on the MATH500 dataset, while Bespoke-Stratos-7B shows a larger increase of approximately 1000 tokens. These align with test-time scaling principle (Snell et al., 2024; Welleck et al., 2024), where the increased response length reflects an enhancement in reasoning capacities.

Bespoke-Stratos-7B, since both reasoning models were fine-tuned on Qwen2.5-7B-Instruct. Since Our data is collected using prompts from Bespoke, which is not align with the prompt format for DeepSeek-7B. In our comparison between SFT and ThinkPO, we maintained consistency by using the same Bespoke-style prompts to evaluate models.

As shown in Table 2 and Figure 4, both models demonstrate an improvement in accuracy across five datasets. For example, Bespoke-Stratos-7B shows an increase in accuracy on all datasets except

3.4 ThinkPO Works for Different-Size Models

Previous experiments are all conducted using a 7B model for training. Now we utilize the Bespoke Stratos dataset and conduct one epoch of SFT training on models of varying sizes within the Qwen2.5 series (Qwen2.5-3B, Qwen2.5-7B, and Qwen2.5-14B). The learning rate is set to 3e-5, and other hyperparameters are kept consistent with Bespoke-Stratos-7B, ensuring the models’ performances. The results after SFT and ThinkPO are presented in Table 3 and Figure 6. First, as the

model scale increases, its accuracy improves across all the datasets after SFT, which aligns with expectations. After applying ThinkPO, all models, regardless of size, achieve further improvements. Specifically, on Math500, all three models show an accuracy increase of 1%–2%. After applying ThinkPO, the Qwen2.5-3B model achieves accuracy improvements across all five datasets, while Qwen2.5-7B and 14B models show improvements on four datasets, which shows that ThinkPO is effective across different model scales, further validating its generalizability and robustness.

4 Ablation

4.1 Whether ThinkPO is Useful when SFT with Short Reasoning Data?

In our previous experiments, we fully fine-tuned the model using long reasoning datasets before applying ThinkPO to further enhance its reasoning ability. However, an important question arises: If we use short reasoning data instead of long reasoning data during the full fine-tuning stage, can Thinking Preference Optimization still improve the model’s reasoning performance effectively?

To investigate this, we conduct the following experiment. We use Qwen2.5-7B as the base model and select a dataset from AI-MO/NuminaMath-CoT(LI et al., 2024) that matches the Bespoke-Stratos dataset with the same data size for fine-tuning. Unlike our previous experiments, the fine-tuning data here consists of short reasoning examples rather than long reasoning ones. Consequently, the fine-tuned model is expected to underperform compared to models trained on long-reasoning data. To equip models with basic reasoning ability, we fine-tune them for three epochs and set learning rate as $1e-5$. Following this, we apply Thinking Preference Optimization using the same dataset in the previous experiments, aiming to further enhance and improve the model’s reasoning performance.

As shown in Table 4, even after fine-tuning on short-reasoning data, ThinkPO still effectively improves the model’s reasoning ability. For example, on the Math500 dataset, after applying ThinkPO, the model’s accuracy improves by approximately 2%. This result demonstrates that models trained with reasoning-style datasets, regardless of response length, can benefit from ThinkPO to enhance and improve their reasoning capability.

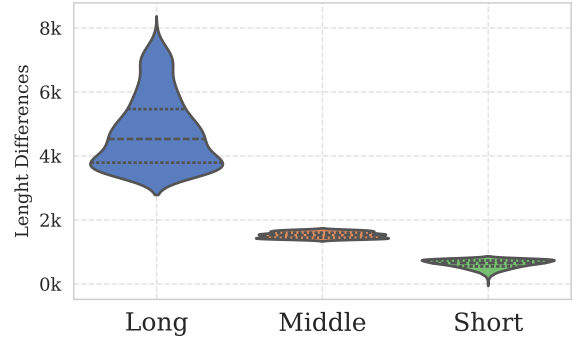


Figure 7: Length difference distribution between chosen and rejected samples across three datasets. These three datasets are 1000 samples selected based on the length difference from our ThinkPO-Dataset. The long dataset exhibits the widest distribution of length differences, while the middle and short datasets have smaller differences with lower mean values and variances.

Table 5: Model performance across three datasets with varying chosen and rejected sample length difference distributions. “Avg Differences” represents the average length difference between chosen and rejected samples. *Short* yields the best overall performance, suggesting that appropriate length differences improve ThinkPO learning, while too large differences may hinder it.

	Short	Middle	Long
Avg Differences	621	1525	4758
MATH500	84.2	81.8	84.0
AIME	26.7	13.3	16.7
GPQA	40.9	41.9	38.9
GSM8K	92.9	92.9	93.0
Olympiad	46.1	45.9	45.9

4.2 Exploring the Impact of Length Differences between Chosen and Rejected Samples on ThinkPO.

In the entire ThinkPO dataset, we select long reasoning data as chosen and short reasoning data as rejected. A key question is whether the length disparity between chosen and rejected samples affects the ThinkPO training because length disparity is not distributed evenly in the dataset. To investigate this, we conduct an experiment to verify the impact of length differences on the ThinkPO training.

The ThinkPO dataset contains approximately 10,000 samples, but the length disparity between chosen and rejected samples is not uniformly distributed. Therefore, we select three datasets with different length distributions: short, middle, and long, each containing 1,000 samples. Figure 7 shows details of the length differences distributions

between chosen and rejected samples in these three datasets, with the long dataset exhibiting the largest and most widely distributed differences, the middle dataset showing moderate differences, and the short dataset having the smallest differences.

Table 5 displays the results after ThinkPO for one epoch, using the Bespoke-Stratos-7B model as the base model. Each dataset shows certain advantages across the five test datasets. However, the short dataset yields the best performance on overall datasets. We propose that when the length difference is smaller, the model’s output distributions for both samples are more consistent, which benefits ThinkPO learning. On the other hand, when it is too large, it may not help the model’s learning.

5 Related Works

LLM Reasoning Ability. With the development of large models, reasoning ability (Wang et al., 2022; Zhang et al., 2023; Yao et al., 2023; Plaat et al., 2024) has become one of the most crucial capabilities and a necessary condition for achieving AGI (Artificial General Intelligence) (Minaee et al., 2024; Xu et al., 2024; Morris et al., 2023; Feng et al., 2024; Krishnan, 2025). The earliest appearance of long-chain reasoning ability in large models can be traced to OpenAI o1 (Jaech et al., 2024; Arrieta et al., 2025; Hurst et al., 2024), which excelled across various mathematical reasoning test sets and outperform contemporary LLMs.

This was followed by the release of the QwQ model (Yang et al., 2024b; Bai et al., 2023a,b; Chu et al., 2024), which trained reasoning capabilities using a process reward model approach (Li and Li, 2024; Ma et al., 2023; Zhang et al., 2025; Lambert et al., 2024). Currently, the emergence of DeepSeek R1 (DeepSeek-AI et al., 2025) and Kimi 1.5 (Team et al., 2025) has further enhanced the reasoning abilities of large open-source models. DeepSeek R1 utilizes a simple rule-based reward model (Ramesh et al., 2024; Hu, 2025; Shao et al., 2024; Alonso et al., 2025; Kirk et al., 2023; Yang et al., 2024a) to effectively boost the model’s reasoning performance, bringing about an aha moment that narrows the reasoning capability gap between open-source and closed-source models. On the other hand, Kimi 1.5 employs several tricks, such as long-to-short reasoning, to achieve high efficiency in LLM reasoning performance.

Many works on open-source reasoning models have also emerged. First is Sky-Thought T1 (Team,

2025a), which uses QwQ-32B-Preview as a teacher model to generate reasoning answers for training data. Then, Bespoke-Stratos (Labs, 2025) built upon Sky-Thought T1, using DeepSeek R1 as the teacher model to generate answers for Sky-Thought data. Since DeepSeek R1 has far superior reasoning abilities compared to QwQ-32B-Preview, the generated data quality is higher, allowing Bespoke-Stratos-7B and Bespoke-Stratos-32B models to achieve DeepSeek-level advanced reasoning performance after training on around 17k data points. Recently, s1 (Muennighoff et al., 2025) and LIMO (Ye et al., 2025) have emphasized that fine-tuned, high-quality data construction is essential for models to achieve SOTA reasoning capabilities.

Direct Preference Optimization. RLHF (Chaudhari et al., 2024; Kirk et al., 2023; Kaufmann et al., 2023) is designed to align model outputs with human preferences after supervised fine-tuning (SFT). Various methods have been introduced, such as Proximal Policy Optimization (PPO) (Engstrom et al., 2019; Huang et al., 2022; Wijmans et al., 2019). However, PPO is an online method that requires significant computational resources. To address this, Direct Preference Optimization was proposed, enabling offline training with only chosen and rejected sample pairs while reducing computational costs compared to PPO. Recently, several DPO variants (Wu et al., 2024b,a; Qi et al., 2024; Zhong et al., 2024; Su et al., 2025) have emerged, including StepDPO (Lai et al., 2024), KTO (Ethayarajh et al., 2024), SimPO (Meng et al., 2024), LongDPO (Ping et al., 2025), Test-Time Preference Optimization (Li et al., 2025) etc. Among them, LongDPO shares similarities with our proposed method. However, LongDPO primarily focuses on improving long-form story generation instead of reasoning abilities.

6 Conclusion

We propose ThinkPO, a lightweight post-SFT method without additional high-quality long-reasoning data. By treating short responses as rejected and long responses as chosen, ThinkPO encourages models to generate more detailed reasoning. Experiments show that ThinkPO improves accuracy by 8.6% and output length by 25.9%, and boosts DeepSeek-R1-Distill-Qwen-7B’s MATH500 score from 87.4% to 91.2%. These results highlight ThinkPO as an efficient way to enhance reasoning with existing long reasoning data.

Limitations

ThinkPO can further enhance SFT-ed models without requiring additional high-quality long reasoning data. However, since ThinkPO is based on the DPO method, it is sensitive to hyperparameters, requiring careful tuning of β and learning rate to achieve optimal improvements.

References

Noguer I Alonso and 1 others. 2025. The mathematics of group relative policy optimization: A multi-agent reinforcement learning approach. *The Mathematics of Group Relative Policy Optimization: A Multi-Agent Reinforcement Learning Approach (January 03, 2025)*.

Aitor Arrieta, Miriam Ugarte, Pablo Valle, José Antonio Parejo, and Sergio Segura. 2025. o3-mini vs deepseek-r1: Which one is safer? *arXiv preprint arXiv:2501.18438*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023a. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023b. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. 2024. RLhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *arXiv preprint arXiv:2404.08555*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, and 1 others. 2024. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2019. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Tao Feng, Chuanyang Jin, Jingyu Liu, Kunlun Zhu, Haoqin Tu, Zirui Cheng, Guanyu Lin, and Jiaxuan You. 2024. How far are we from agi. *arXiv preprint arXiv:2405.10313*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. *Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems*. *Preprint*, arXiv:2402.14008.

Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.

Shengyi Huang, Anssi Kanervisto, Antonin Raffin, Weixun Wang, Santiago Ontañón, and Rousslan Fernand Julien Dossa. 2022. A2c is a special case of ppo. *arXiv preprint arXiv:2205.09123*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. 2023. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Vinod Krishnan. 2025. From artificial intelligence (ai) to artificial general intelligence (agi)—the road ahead. *Journal of the World Federation of Orthodontists*, 14(1):1–2.

- Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation. Accessed: 2025-01-22.
- Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, and 1 others. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.
- Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. [<https://huggingface.co/AI-M0/NuminaMath-CoT>](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).
- Wendi Li and Yixuan Li. 2024. Process reward model with q-value rankings. *arXiv preprint arXiv:2410.11287*.
- Yafu Li, Xuyang Hu, Xiaoye Qu, Linjie Li, and Yu Cheng. 2025. Test-time preference optimization: On-the-fly alignment via iterative textual feedback. *Preprint*, arXiv:2501.12895.
- 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*.
- Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let’s reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.
- Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.
- Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.
- Meredith Ringel Morris, Jascha Sohl-Dickstein, Noah Fiedel, Tris Warkentin, Allan Dafoe, Aleksandra Faust, Clement Farabet, and Shane Legg. 2023. Levels of agi: Operationalizing progress on the path to agi. *arXiv preprint arXiv:2311.02462*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.
- Bowen Ping, Jiali Zeng, Fandong Meng, Shuo Wang, Jie Zhou, and Shanghang Zhang. 2025. Longdpo: Unlock better long-form generation abilities for llms via critique-augmented stepwise information. *arXiv preprint arXiv:2502.02095*.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*.
- Biqing Qi, Pengfei Li, Fangyuan Li, Junqi Gao, Kaiyan Zhang, and Bowen Zhou. 2024. Online dpo: Online direct preference optimization with fast-slow chasing. *arXiv preprint arXiv:2406.05534*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.
- Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. 2024. Group robust preference optimization in reward-free rlhf. *arXiv preprint arXiv:2405.20304*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Xuerui Su, Yue Wang, Jinhua Zhu, Mingyang Yi, Feng Xu, Zhiming Ma, and Yuting Liu. 2025. Reveal the mystery of dpo: The connection between dpo and rl algorithms. *arXiv preprint arXiv:2502.03095*.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chunying Tang, Congcong Wang, Dehao Zhang, Enming Yuan,

- Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. [Kimi k1.5: Scaling reinforcement learning with llms](#). *Preprint*, arXiv:2501.12599.
- NovaSky Team. 2025a. Sky-t1: Train your own o1 preview model within \$450. <https://novasky-ai.github.io/posts/sky-t1>. Accessed: 2025-01-09.
- OpenThoughts Team. 2025b. Open Thoughts. <https://open-thoughts.ai>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.
- Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilya Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*.
- Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. 2019. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*.
- Junkang Wu, Xue Wang, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. 2024a. α -dpo: Adaptive reward margin is what direct preference optimization needs. *arXiv preprint arXiv:2410.10148*.
- Junkang Wu, Yuexiang Xie, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. 2024b. β -dpo: Direct preference optimization with dynamic β . *arXiv preprint arXiv:2407.08639*.
- Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, and 1 others. 2024. A survey of resource-efficient llm and multimodal foundation models. *arXiv preprint arXiv:2401.08092*.
- Adam X Yang, Maxime Robeyns, Thomas Coste, Zhengyan Shi, Jun Wang, Haitham Bou-Ammar, and Laurence Aitchison. 2024a. Bayesian reward models for llm alignment. *arXiv preprint arXiv:2402.13210*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. [Limo: Less is more for reasoning](#). *Preprint*, arXiv:2502.03387.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.
- Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multi-modal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*.
- Han Zhong, Guhao Feng, Wei Xiong, Xinle Cheng, Li Zhao, Di He, Jiang Bian, and Liwei Wang. 2024. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*.

A Appendix

A.1 Evaluating ThinkPO with Different Temperatures

In our experiments, we initially evaluated the model at a temperature of 0.7. While this provides a good measure of performance, it is important to explore different sampling conditions for a more robust analysis. Therefore, we additionally tested temperatures of 0.1 and 0.5 to examine how ThinkPO impacts Bespoke-Strato-7B under varying levels of randomness in sampling. By comparing results across these temperature settings, we can assess whether ThinkPO consistently enhances the model’s reasoning ability regardless of generation strategy. To provide a comprehensive evaluation, we average the results across all three temperatures. The results are shown in Table 6.

Our findings demonstrate that ThinkPO consistently improves model performance across different temperature settings. Specifically, at temperatures of 0.1 and 0.7, accuracy increases on four datasets, while at 0.5, improvements are observed on three. To gain a more holistic understanding of ThinkPO’s impact, we average the results across all temperature settings, showing that ThinkPO enhances performance on all five datasets. Notably, on MATH500, ThinkPO improves accuracy by 1.4%. These results further validate the effectiveness of our proposed method and demonstrate its ability to consistently enhance reasoning performance across different sampling conditions.

A.2 Analysis of our Reproduce Model in other datasets

Previously, we only presented the changes in accuracy, average response length, and reasoning-supportive words count over training steps on the MATH500 dataset. Here, we extend our analysis by showcasing results on two additional datasets (like GSM8K) from our reproduced model. The detailed results are illustrated in Figure 9.

As observed in the results for GSM8K and Olympiad Bench Math, the model exhibits a similar trend to MATH500 across all three metrics. During the early stages of SFT, the model’s reasoning ability improves rapidly. However, in later stages, it reaches a performance plateau. ThinkPO effectively helps the model overcome this bottleneck, further enhancing its reasoning capability.

A.3 Details of Data Curation

Here, we show the details of how to collect the ThinkPO Dataset. we collect it in the following manner, referring to (Team et al., 2025): For each question q in $\mathcal{D}_{sft}$, we use Qwen2.5-Math-7B-Instruct (Yang et al., 2024b) to generate a short reasoning response o_{short} , pairing it with the long reasoning response o_{long} in $\mathcal{D}_{sft}$. We then retain the samples where Qwen2.5-Math-7B-Instruct’s answer matched DeepSeek R1’s answer, resulting in 8,080 samples. Additionally, we include 2,000 samples where Qwen2.5-Math-7B-Instruct’s answer differed from DeepSeek R1’s but adhered to the correct response format, including more output distribution in $\mathcal{D}_{dpo}$. All of these combined samples consequently form the final dataset $\mathcal{D}_{dpo}$.

A.4 Training Recipe

Here, we provide the corresponding hyperparameters—batch size, learning rate, and β —that were used to achieve these optimal outcomes. All the hyperparameters are presented in Table 7.

Besides, we present the training loss curves, gradient norm curves, and margin curves for three models during the ThinkPO phase in Figure 5. These metrics provide insights into how the models perform throughout the training process, including their convergence behavior, stability of gradients, and the differences in preference between chosen and rejected samples. By examining these curves, we can better understand the effectiveness of ThinkPO in enhancing model performance.

A.5 Examples of LLM’s outputs before and after ThinkPO

We present the changes in the total number of reasoning-supportive words (such as wait, hmm, let’s think, etc.) throughout both the SFT and ThinkPO training stages in Figure 2 and Figure 9. These words serve as indicators of the model’s reasoning process, reflecting its ability to structure logical steps before arriving at a final answer. Our results show that the number of reasoning-supportive words increases significantly during the initial stages of SFT but eventually plateaus, suggesting that conventional fine-tuning alone may not be sufficient to further enhance structured reasoning. However, after applying ThinkPO, we observe a clear upward trend in the use of these reasoning-supportive expressions, indicating that our method effectively encourages the model to adopt a more

Table 6: Evaluation of Bespoke-Strato-7B with different temperatures(0.1,0.5,0.7). Across different values of temperatures, the model achieves accuracy improvements on most datasets. After averaging the results, ThinkPO consistently enhances the model’s performance across all five datasets.

	Temperature=0.1		Temperature=0.5		Temperature=0.7		Average		
	+SFT	+ThinkPO	+SFT	+ThinkPO	+SFT	+ThinkPO	+SFT	+ThinkPO	Improv.
MATH500	70.2	73.4 ↑	81.4	82.6 ↑	84.0	82.8 ↓	78.5	79.6 ↑	1.4%
AIME	10.0	16.7 ↑	20.0	16.7 ↓	20.0	23.3 ↑	16.7	18.9 ↑	13.2%
GPQA	34.9	30.8 ↓	33.8	41.0 ↑	37.9	43.4 ↑	35.5	38.4 ↑	8.1%
GSM8K	89.3	91.0 ↑	92.4	92.3 ↓	92.9	93.3 ↑	91.5	92.2 ↑	0.7%
Olympiad	32.8	39.6 ↑	42.3	44.8 ↑	44.1	48.5 ↑	39.7	44.3 ↑	11.6%

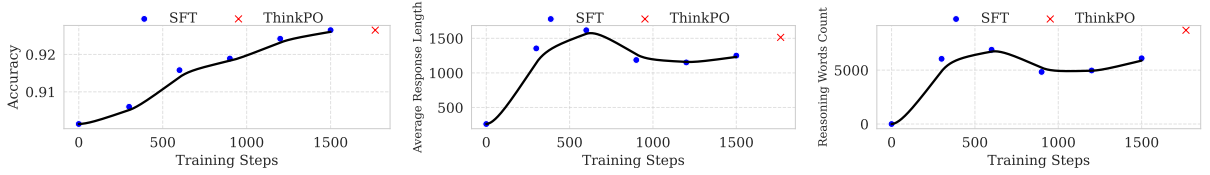


Figure 8: Analysis of accuracy(Left), average response length(Middle) and reasoning-supportive words count(Right, like wait, hmm, etc) in reproducing Bespoke-Stratos-7B. We evaluate the model on GSM8K every 300 steps and record results. In the early training stages, all of them improve significantly. However, in the later stages (e.g., after 1200 steps), the model’s performance plateau. When ThinkPO is applied, we see additional improvements in all of the three aspects, demonstrating the effectiveness of Think Preference Optimization.

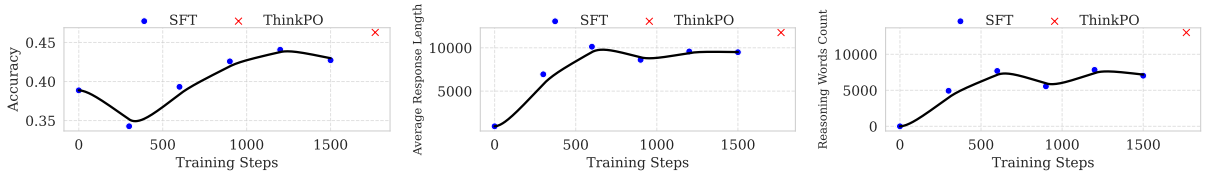


Figure 9: Analysis of accuracy(Left), average response length(Middle) and reasoning-supportive words count(Right, like wait, hmm, etc) in reproducing Bespoke-Stratos-7B. We evaluate the model on OlympiadBenchMath every 300 steps and record results. In the early training stages, all of them improve significantly. However, in the later stages (e.g., after 1200 steps), the model’s performance plateau. When ThinkPO is applied, we see additional improvements in all of the three aspects, demonstrating the effectiveness of Think Preference Optimization.

deliberative reasoning process.

We provide examples of model outputs before and after applying ThinkPO in Table 12 and Table 13. Before ThinkPO, the model’s responses tend to be more direct, with fewer reasoning-supportive words, often resulting in incorrect or incomplete answers. In contrast, after applying ThinkPO, the model generates responses that utilize a greater number of reasoning-supportive words. This shift leads to a noticeable improvement in answer correctness, reinforcing the effectiveness of ThinkPO in enhancing the model’s reasoning ability. These findings highlight that ThinkPO not only improves accuracy but also aligns the model’s output with human-like problem-solving patterns.

A.6 Validation of ThinkPO on OpenR1 Datasets and Deepseek-style metric

To further validate the effectiveness of ThinkPO, we conduct experiments using a newly constructed dataset and a more robust evaluation metric.

First, we adopt the open-source dataset **OpenR1-Math-220K**, which is created by re-collecting DeepSeek-R1 responses for the questions from NuminaMath-CoT. We randomly sample 15K examples from the dataset, where the DeepSeek-R1 responses serve as long *chosen* samples, and the original NuminaMath-CoT responses serve as short *rejected* samples.

Second, we introduce a **DeepSeek-style metric**, where the model is prompted to generate five responses per question, and the final accuracy is computed across all generated responses. This met-

Table 7: The optimal hyperparameters identified in our experiments are listed here, including batch size, learning rate (lr), and beta. These parameters were carefully tuned to achieve the best performance improvements.

	Deepseek-7B	Bespoke-7B	Bespoke-7B-reproduced
batch size	48	48	48
lr	1e-7	5e-7	3e-7
β	0.01	0.01	0.01
	Qwen2.5-3B-SFT	Qwen2.5-7B-SFT	Qwen2.5-14B-SFT
batch size	48	48	48
lr	5e-7	8e-8	1e-7
β	0.01	0.01	0.01

Table 8: Results of baseline DPO and ThinkPO on the model. DeepSeek-R1 responses are used as *chosen* samples and the model outputs after SFT are collected as *rejected* samples.

	Baseline +DPO	Our +ThinkPO	Improv. %
MATH500	84.6	84.6	0.0%
AIME22-24	22.0	22.4	2.0%
GPQA	42.5	43.0	1.2%
GSM8K	93.2	93.0	-0.2%
Olympiad	48.6	49.1	1.0%
Average	58.2	58.4	0.4%

DeepSeek-R1 responses are used as *chosen* samples and the model outputs after SFT are collected as *rejected* samples. Table 8 shows that the baseline performs comparably to ThinkPO. However, collecting rejected samples via model inference introduces additional time and resource costs. In contrast, ThinkPO leverages pre-existing datasets to extract long *rejected* samples efficiently, often with longer output lengths, demonstrating the method’s efficiency and practicality.

ric provides more stable and reliable evaluation compared to single-response accuracy.

We conduct the following experiments:

1. We first fine-tune Qwen-2.5-7B-Instruct on the 15K subset of OpenR1-Math-220K. Then, we further improve it using ThinkPO. As shown in Table 8, both the model’s accuracy and output length improve after applying ThinkPO.
2. We apply ThinkPO training to two open-source models: open-r1/OpenR1-Qwen-7B and deepseek-ai/DeepSeek-R1-Distill-Qwen-7B. The results in Table 9 show that both models benefit from further accuracy gains after ThinkPO.
3. We extend the study to the Qwen-2.5-Instruct family with three different model sizes: 3B, 7B, and 14B. All models are trained with standard SFT and then improved via ThinkPO. Table 9 shows consistent performance improvements across scales.
4. Finally, we introduce a **baseline**, where

Table 9: Accuracy and Average Response Length comparison for our finetuned Qwen-2.5-7B-Instruct before and after ThinkPO. The "Improv." column shows the percentage change of **Ours** over the model. After applying ThinkPO, its accuracy and length almost improve across datasets, further validating the effectiveness of ThinkPO.

Dataset	Accuracy				Average Response Length			
	Base	+SFT	+ThinkPO	Improv.(%)	Base	+SFT	+ThinkPO	Improv.(%)
MATH500	74.0	84.5	84.6	0.1%	637	5416	5521	1.9%
AIME22-24	10.0	20.7	22.4	8.2%	942	17828	18082	1.4%
GPQA	34.9	41.8	43.0	2.9%	12	10096	10449	3.5%
GSM8K	90.1	92.8	93.0	0.2%	260	1599	1606	0.4%
Olympiad	38.9	48.3	49.1	1.7%	942	10876	10975	0.9%
Avg.	49.6	57.6	58.4	1.4%	558	9117	9249	1.4%

Table 10: Accuracy and Average Response Length comparison for OpenR1-Qwen-7B and DeepSeek-R1-Distill-Qwen-7B before and after ThinkPO. The "Improv." column shows the percentage change of **Ours** over the base model.

OpenR1-Qwen-7B						
Dataset	Accuracy			Average Response Length		
	OpenR1 (SFT)	Ours (+ThinkPO)	Improv. (%)	OpenR1 (SFT)	Ours (+ThinkPO)	Improv. (%)
MATH500	90.7	90.6	−0.1%	3276	3427	4.6%
AIME22-24	43.1	44.2	2.6%	13559	13764	1.5%
GPQA	42.2	43.9	4.0%	9533	9972	4.6%
GSM8K	95.0	95.1	0.1%	1227	1274	3.8%
Olympiad	60.2	60.5	0.5%	8587	8594	0.1%

DeepSeek-R1-Distill-Qwen-7B						
Dataset	Accuracy			Average Response Length		
	DeepSeek (SFT)	Ours (+ThinkPO)	Improv. (%)	DeepSeek (SFT)	Ours (+ThinkPO)	Improv. (%)
MATH500	91.0	91.8	0.9%	3702	3832	3.5%
AIME22-24	48.1	48.0	−0.2%	13867	13918	0.4%
GPQA	47.4	48.7	2.7%	6113	6450	5.5%
GSM8K	86.7	87.1	0.5%	467	464	−0.6%
Olympiad	60.2	61.1	1.5%	8541	8525	−0.2%

Table 11: Results of Models with Different Sizes (3B, 7B, 14B) on the Qwen-2.5 Family. We evaluate models of different sizes (3B, 7B, 14B) trained with Supervised Fine-Tuning (SFT) and Think Preference Optimization (ThinkPO). Models are fine-tuned on the Bespoke-Strato-Dataset for 1 epoch. As model size increases, accuracy improves across all five test datasets. After ThinkPO training, accuracy improves consistently for models of all sizes, including the smallest (3B), demonstrating that ThinkPO enhances reasoning ability across different model scales.

	Qwen 2.5-3B-Instruct			Qwen 2.5-7B-Instruct			Qwen 2.5-14B-Instruct		
	+SFT	+ThinkPO	Improv.	+SFT	+ThinkPO	Improv.	+SFT	+ThinkPO	Improv.
MATH500	64.9	65.6	1.2%	84.5	84.6	0.1%	90.7	91.2	0.5%
AIME22-24	7.1	9.3	30.8%	20.7	22.4	8.2%	44.0	43.6	−0.9%
GPQA	29.5	29.9	1.4%	41.8	43.0	2.9%	55.3	55.6	0.6%
GSM8K	83.5	84.8	1.6%	92.8	93.0	0.2%	94.8	95.0	0.2%
Olympiad	28.7	28.6	−0.3%	48.3	49.1	1.7%	58.7	59.2	0.8%

Question	What is the domain of the function $f(x) = \frac{2-x}{\log(2-\log(x-2))}$, where $\log$ is the base 10 logarithm function? Express your answer in interval notation.
SFT	Okay, so I need to find the domain of the function $f(x) = (2 - x)/\log(2 - \log(x - 2))$. hmm, let's start by recalling that the domain of a function includes all the real numbers x for which the function is defined. Since this function has a logarithm in the denominator, there are a couple of things I need to check: first, the argument of the outer logarithm (the one in the denominator) must be positive because logarithms are only defined for positive numbers. Second, the argument of the inner logarithm (the one inside the denominator's argument) must also be positive because we can't take the logarithm of a non-positive number. Let me break this down step by step.... But wait, 102 is already the upper bound, so actually, the domain is (2, 102). But we have to make sure that within (2, 102), the expression $2 - \log(x - 2)$ is positive, which we already considered. Let me verify this again. So, starting with the inner logarithm: $x > 2$. Then, $\log(x - 2)$ is defined. Then, the outer logarithm requires that $2 - \log(x - 2) > 0$, which as we saw, leads to $x < 102$. So combining $x > 2$ and $x < 102$ gives (2, 102). Additionally, we need to ensure that the denominator is not zero, which occurs at $x = 102$, but since 102 is already the upper limit, the interval is open at 102. Therefore, the domain is (2, 102). Let me check if there are any other restrictions or if I missed something. wait, another thing to consider: the numerator is $2 - x$. The numerator is a linear function, so it's defined for all real numbers. The denominator is $\log(2 - \log(x - 2))$, ... Then, $\log(3 - 2) = \log(1) = 0$. Then, $2 - 0 = 2$, so $\log(2)$ is the denominator. That's valid. If $x = 100$, then $\log(100 - 2) = \log(98)$, which is positive, so $2 - \log(98)$ would be $2 - \log(98)$, which is positive because $\log(98)$ is less than 2 (since $10^2 = 100$). So, denominator is positive. If x approaches 2 from the right, x approaches $2+$, so $x - 2$ approaches $0+$, so $\log(x - 2)$ approaches $-\infty$, so $2 - \log(x - 2)$ approaches $+\infty$, so denominator is defined. If x approaches 102 from the left, x approaches $102-$, so $x - 2$ approaches $100-$, so $\log(x - 2)$ approaches $\log(100-)$, which is close to 2, so $2 - \log(x - 2)$ approaches 0 from the positive side, so the denominator approaches $0+$, making the whole function approach $+\infty$. But x can't be 102. So, in all other points between 2 and 102, the denominator is defined and positive, so the function is defined there. Therefore, the domain is indeed (2, 102). I think that's it. Final Answer The domain of the function is (2, 102).
ThinkPO	Okay, so I need to find the domain of the function $f(x) = (2 - x)/\log(2 - \log(x - 2))$, where the $\log$ here is base 10. hmm, let me think step by step about what's required here. ... But wait, I need to check if this $x > 2$ is compatible with the first condition. So maybe I need to solve both inequalities together? Let me start with the second condition: $\log(x - 2)$ is defined when $x - 2 > 0$, so $x > 2$. So that's the domain from the argument of the inner logarithm. But then, ... So combining $x > 2$ and $x < 102$ gives me $2 < x < 102$. wait, but hold on. Let me verify this. If $\log(x - 2) < 2$, then $x - 2 < 10^2 = 100$, so $x < 102$. But since $x - 2$ must be positive (because $\log$ requires positive argument), $x > 2$. So indeed, x is in (2, 102). But wait, when I take $\log(x - 2) < 2$, ... If x approaches 2 from the right, $x - 2$ approaches $0+$, so $\log(x - 2)$ approaches $-\infty$, so $2 - \log(x - 2)$ approaches $+\infty$. wait, but as x approaches $2+$, the inner $\log(x - 2)$ goes to $-\infty$, so $2 - \log(x - 2)$ goes to $+\infty$. Therefore, the denominator $\log(2 - \log(x - 2))$ approaches $\log(+\infty)$ which is $+\infty$. wait, but that would mean the denominator is positive and approaching infinity. But wait, is there a point where the denominator could be zero? Let's check. For $2 - \log(x - 2) = 0$, then $\log(x - 2) = 2$, so $x - 2 = 10^2 = 100$, so $x = 102$. But $x = 102$ is the upper bound of the domain. So at $x = 102$, the denominator is $\log(2 - \log(100)) = \log(2 - 2) = \log(0)$, which is undefined. So x cannot be 102. So the domain is (2, 102). But wait, we also have to consider the numerator. The numerator is $2 - x$. So $2 - x$ can be zero, but if the denominator is non-zero, then the function is defined. If numerator is zero, then $f(x) = 0/\text{denominator}$, which is zero. But division by zero is undefined. wait, but if numerator is zero, does that make the entire expression zero, but with denominator non-zero? For example, if $x = 2$, then numerator is 0, denominator is $\log(2 - \log(0))$. wait, $x=2$ is not in the domain because $x - 2 = 0$, so $\log(0)$ is undefined. ... wait, but the denominator is $\log(2 - \log(x - 2))$. So the denominator is $\log$ of something. The logarithm output can be any real number, but in this case, the denominator is used in the function, so as long as the argument inside the log is positive, which we already considered. But the denominator itself can be any real number except zero. wait, no. wait, the denominator is $\log(2 - \log(x - 2))$, wait, so... Let's compute $\log(10)$ is 1 (base 10), so $2 - 1 = 1$, $\log(1) = 0$, ... That seems correct. wait, but let's check another value. If x is 11, then $x - 2 = 9$, $\log(9) = 0.954$, $2 - 0.954 = 1.046$, so $\log(1.046)$ is a positive number. So denominator is defined and non-zero. If x is 13, then $x - 2 = 11$, $\log(11) = 1.041$, $2 - 1.041 = 0.959$, $\log(0.959)$ is negative. But wait, the original function is defined only if the denominator is $\log(\text{positive number})$. So if the denominator is $\log$ of a number less than 1, that's negative, but is that allowed? wait, no. wait, the denominator is $\log(2 - \log(x - 2))$. For the log to be defined, $2 - \log(x - 2)$ must be positive. So if $2 - \log(x - 2)$ is negative, then $\log$ of that is undefined (since $\log$ of a negative number is not real). wait, $x=1$ is less than 2, ... wait, no. wait, the denominator is $\log(2 - \log(x - 2))$. For $\log$ to be defined, the argument must be positive. So if $2 - \log(x - 2)$ is positive, then $\log$ is defined, regardless of whether it's positive or negative. wait, no. wait, So intervals (2,12) and (12,102). That should be correct.

Table 12: One example of LLM's wrong outputs before ThinkPO and correct outputs after ThinkPO. ThinkPO helps models generate more reasoning-supportive discourse markers, such as wait, hmm, and other hesitation cues.

Question	How many positive two-digit integers are factors of both 100 and 150?
SFT	Okay, so I need to figure out how many positive two-digit integers are factors of both 100 and 150. hmm, let's start by understanding the problem....So, the GCD of 100 and 150 is 50. That means all the common factors of 100 and 150 are the factors of 50. Let me list out the factors of 50 to make sure. The factors of 50 are 1, 2, 5, 10, 25, and 50. wait, that's six factors. But the problem asks for two-digit integers. From this list, the two-digit factors would be 25 and 50. So that's two numbers. But wait, hold on. Let me verify if 25 and 50 are indeed factors of both 100 and 150. For 25: $100 / 25 = 4$, which is an integer. $150 / 25 = 6$, which is also an integer. So yes, 25 is a common factor. For 50: $100 / 50 = 2$, and $150 / 50 = 3$. Both integers, so 50 is also a common factor. But wait, are there any other two-digit numbers that are factors of both 100 and 150? Let's check. The factors of 100 are 1, 2, 4, 5, 10, 20, 25, 50, 100. The factors of 150 are 1, 2, 3, 5, 6, 10, 15, 25, 30, 50, 75, 150. The common factors are 1, 2, 5, 10, 25, 50. So, the two-digit common factors are 25 and 50. That gives us two numbers. wait, but hold on. The problem says "positive two-digit integers". ... So, factors of 50 are 1, 2, 5, 10, 25, 50. So, two-digit ones are 25 and 50. Therefore, the answer is 2. But wait, let me check if there are any other numbers between 10 and 99 that divide both 100 and 150. For example, 10 divides both. 10 is two-digit. 20 divides 100 but $150 / 20 = 7.5$, which is not integer. So 20 is out. 30 divides 150 but $100 / 30$ is approximately 3.333, not integer. 15 divides 150 but $100 / 15$ is about 6.666, not integer. 75 divides 150 but $100 / 75$ is 1.333, not integer. So, indeed, only 25 and 50 are in the two-digit range. Therefore, the answer is 2. But wait, wait a second. Let me check 25 and 50 again. 25 divides both 100 and 150, yes. 50 divides both, yes. Are there any other numbers between 10 and 99 that divide both? Let me check 10. 10 divides 100 and 150. 10 is two-digit. But we already have 10, 25, 50. wait, 10 is a factor, but it's single-digit. So 10 is not included. So 25 and 50 are the only two. Therefore, the answer is 2. So, the number of positive two-digit integers that are factors of both 100 and 150 is 2. But just to be thorough, let me check 25: $100 / 25 = 4$, $150 / 25 = 6$. Both integers. 50: $100 / 50 = 2$, $150 / 50 = 3$. Both integers. So yes, they are factors. Are there any others? Let me check 30: $100 / 30 = 3.333$, not integer. 40: $100 / 40 = 2.5$, not integer. 45: $100 / 45 = 2.222$, no. 55: $100 / 55 = 1.818$, no. 60: $100 / 60 = 1.666$, no. 65: $100 / 65 = 1.538$, no. 70: $100 / 70 = 1.428$, no. 75: $100 / 75 = 1.333$, no. 80: $100 / 80 = 1.25$, no. 85: $100 / 85 = 1.176$, no. 90: $100 / 90 = 1.111$, no. 95: $100 / 95 = 1.052$, no. So, none of these two-digit numbers are factors of both 100 and 150. Therefore, only 25 and 50. Therefore, the answer is 2. So, the final answer is 2, which in boxed form is 2.
ThinkPO	Okay, so I need to figure out how many positive two-digit integers are factors of both 100 and 150. hmm, let's start by understanding the problem.... For prime 5, both have exponent 2, so $\min(2,2)=2$. Prime 3 is only in 150, so we ignore it. So GCD is $2^2 * 5^2 = 2 * 25 = 50$. wait, so the GCD of 100 and 150 is 50. ... wait, that's two numbers. But let me verify. wait, 50's divisors: 1, 2, 5, 10, 25, 50. Yep, that's all. So two-digit numbers here are 10 and 50. So the answer would be 2? But let me make sure I didn't miss anything. But wait, is 10 and 50 the only two-digit divisors of 50? Let's check. 50 divided by 10 is 5, which is integer. 50 divided by 50 is 1. So yes, those are the divisors. But wait, 5 is a single-digit number. So the two-digit divisors are 10 and 50. So that's two numbers. wait, but the problem says positive two-digit integers; so 10 and 50. But wait, is 50 a two-digit number? Yes, 50 is two digits. But 10 is also two digits. So that's two numbers. So the answer is 2? hmm, but let me cross-verify this. Alternatively, maybe I should approach it another way. ... So from 10 to 99, the numbers that divide both 100 and 150 are 10, 25, and 50. wait, but 25 and 50 are two-digit numbers. So that's three numbers: 10, 25, 50. wait, but why did I think earlier that 25 is a two-digit number? It is. So three numbers. But then 10, 25, 50. So three numbers. But when I first thought of the GCD approach, I considered the divisors of 50, which are 1, 2, 5, 10, 25, 50. So only two two-digit divisors. But here, when checking manually, I found three. So there's a discrepancy. Which one is correct? wait, perhaps I made a mistake in the initial approach. Let's check again. The GCD of 100 and 150 is 50. So the common factors are the divisors of 50. Divisors of 50 are 1, 2, 5, 10, 25, 50. So yes, two-digit divisors are 10 and 50. So that's two numbers. But according to the manual check, there are three. So which is correct? wait, let's check 25. 25 is a divisor of both 100 and 150. 25 divides 100 because $25*4=100$, and 25 divides 150 because $25*6=150$. So 25 is indeed a common factor. But 25 is a two-digit number, right? 2 and 5. So yes, two digits. So 25 is a two-digit divisor. Then 10 and 50. So that's three numbers. But according to the GCD method, the common factors are exactly the divisors of 50, which are six numbers, but only two two-digit. However, when I checked manually, I found three two-digit divisors. So where's the mistake? wait, maybe I made a mistake in the GCD calculation. Let me check again. 100 factors: $2^2 * 5^2$. 150 factors: $2^1 * 3^1 * 5^2$. Common prime factors: $2^{\min(2,1)} * 5^{\min(2,2)} = 2^1 * 5^2 = 2 * 25 = 50$. So that's correct. So the common factors are exactly the divisors of 50. Therefore, divisors of 50 are 1, 2, 5, 10, 25, 50. So the two-digit ones are 10, 25, 50. wait, that's three numbers. But according to GCD approach, divisors of GCD. So why the discrepancy? wait, but if the GCD is 50, then the common factors are exactly the divisors of 50. So that's six numbers, but only three are two-digit. So why did the manual check show three? But according to the list, divisors of 50 are 1, 2, 5, 10, 25, 50. So two-digit are 10, 25, 50. That's three. So that's correct. But why did I think earlier that there were two? Because maybe I made a mistake in the first thought process. wait, no. wait, when I first considered the problem, I thought the answer is two. But then when I did the manual check, I found three. So which is correct? Let me verify each of the three numbers: 10: 10 divides 100 ($10*10=100$) and 150 ($10*15=150$). So yes. 25: $25*4=100$, $25*6=150$. Yes, so 25 divides both. 50: $50*2=100$, $50*3=150$. Yes, so 50 divides both. So all three are valid. Therefore, the answer should be 3. But wait, the initial GCD approach said divisors of 50, which are six numbers, but only three are two-digit. However, when I checked the manual approach, I found three. Therefore, the answer is three? But I need to make sure there are no other two-digit numbers that divide both 100 and 150. wait, let's check 30. 100 divided by 30 is 3.333, not integer. But wait, let me check another way. Alternatively, since the GCD is 50, the common factors are the divisors of 50. So divisors of 50 are 1, 2, 5, 10, 25, 50. So two-digit are 10, 25, 50. So three numbers. Therefore, the answer is 3. But why did I initially think it was two? Maybe I confused divisors of 50 with common divisors. But no, the common divisors are exactly the divisors of the GCD. So if GCD is 50, then common divisors are divisors of 50. Therefore, three numbers. wait, but in the initial thought process, I considered that the GCD is 50, so common factors are 1, 2, 5, 10, 25, 50. Then two-digit are 10, 25, 50. Three numbers. Therefore, the answer is 3. But let me check once more. Let's take 25. 25 divides 100 and 150. $25*4=100$, $25*6=150$. Yes. 10: $10*10=100$, $10*15=150$. Yes. 50: $50*2=100$, $50*3=150$. Yes. So all three are valid. Therefore, the number of positive two-digit integers that are factors of both 100 and 150 is 3. But wait, hold on. Let me check 5. 5 is a divisor of both, but it's single-digit. 1, 2, 5 are also divisors but single-digit. So the only two-digit divisors are 10, 25, 50. So three numbers. Therefore, the final answer should be 3. But wait, when I first did the GCD approach, I thought it was two, but after manual checking, I found three. So why the discrepancy? But wait, the initial problem says positive two-digit integers; so numbers from 10 to 99. So 10, 25, 50 are within that range. So yes, three numbers. Therefore, after thorough checking, the answer is 3. So I need to make sure I didn't make a mistake in the initial GCD approach. But since GCD is 50, and common factors are divisors of 50, then three two-digit. So the answer is 3. But let me also check if 50 is indeed a two-digit number. Yes, 50 has two digits. 10 is two-digit. 25 is two-digit. So three numbers. Therefore, I think the correct answer is 3. So the final answer should be 3, boxed as 3

Table 13: One more examples of LLM's wrong outputs before ThinkPO and correct outputs after ThinkPO. ThinkPO helps models generate more reasoning-supportive discourse markers, such as wait, hmm, and other hesitation cues.