Chain of Preference Optimization:
Improving Chain-of-Thought Reasoning in LLMs

Xuan Zhang[*12], **Chao Du**[†1], **Tianyu Pang**[1], **Qian Liu**[1], **Wei Gao**[2], **Min Lin**[1]

[1]Sea AI Lab, Singapore

[2]School of Computing and Information Systems, Singapore Management University

xuanzhang.2020@phdcs.smu.edu.sg; weigao@smu.edu.sg;
{duchao, liuqian, tianyupang, linmin}@sea.com

Abstract

The recent development of chain-of-thought (CoT) decoding has enabled large language models (LLMs) to generate explicit logical reasoning paths for complex problem-solving. However, research indicates that these paths are not always deliberate and optimal. The tree-of-thought (ToT) method employs tree-searching to extensively explore the reasoning space and find better reasoning paths that CoT decoding might overlook. This deliberation, however, comes at the cost of significantly increased inference complexity. In this work, we demonstrate that fine-tuning LLMs leveraging the search tree constructed by ToT allows CoT to achieve similar or better performance, thereby avoiding the substantial inference burden. This is achieved through *Chain of Preference Optimization* (CPO), where LLMs are fine-tuned to align each step of the CoT reasoning paths with those of ToT using the inherent preference information in the tree-search process. Extensive experimental results show that CPO significantly improves LLM performance in solving a variety of complex problems, including question answering, fact verification, and arithmetic reasoning, demonstrating its effectiveness. Our code is available at https://github.com/sail-sg/CPO.

1 Introduction

Recent advances in large language models (LLMs) have shown that constructing reasoning chains is critical to improving their problem-solving capabilities [1, 2, 3, 4, 5, 6, 7]. A representative method is chain-of-thought (CoT) [1], which prompts LLMs to generate intermediate reasoning steps, i.e., thoughts, thereby constructing explicit reasoning paths (as depicted in Figure 1(a)). While straightforward and intuitive, recent research observes that CoT can often overlook optimal reasoning paths and exhibit an unconscious style of answering due to its single-path focus [8, 9]. To foster a more deliberate and conscious reasoning style, Yao et al. [8] propose tree-of-thought (ToT), which generates multiple branching thoughts at each step of the reasoning process and conducts self-evaluation for pruning and planning to search for reasoning paths (as shown in Figure 1(b)). However, despite improving reasoning quality, ToT significantly increases computational complexity, which limits its practical application. This raises the question: Can the strategic depth of ToT be integrated into CoT to enhance its effectiveness while maintaining efficiency?

Existing research has initially provided a positive answer to the above question [10, 11, 12]. A natural strategy is to treat the reasoning path discovered by ToT for each instance as a target for supervision, and then fine-tune LLMs to improve their CoT reasoning abilities [11, 12]. Several methods have been proposed to improve this approach, including using advanced tree-search techniques like Monte
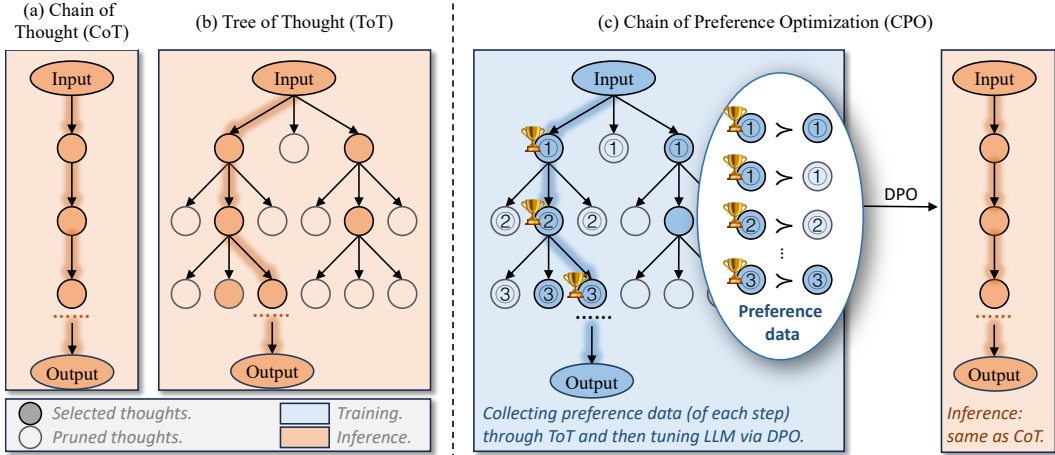

Figure 1: Comparison of CoT, ToT, and CPO methods, where each node illustrates a step in the reasoning process, forming coherent language sequences aimed at solving a problem. The highlighted path indicates the chosen reasoning trajectory. In the CoT method, the LLM generates only one new node at each step, and all generated nodes are used to build the final reasoning path. For ToT, the LLM produces k new nodes at each step, but only the top n-best nodes are kept, with the rest being pruned. In CPO, nodes marked with a trophy represent preferred thoughts, while those marked with numbers are nodes that can be utilized to create preference data. This method uses the search tree structure from ToT to develop paired preference data, subsequently training LLMs to align with these preferences through DPO.

Carlo tree-search (MCTS) and employing external reward models [12, 10] for pruning and planning to gather better reasoning paths as supervision. The effectiveness of these approaches is therefore largely dependent on the quality of the best-discovered reasoning path.

In this paper, we identify a limitation in these approaches: they overlook the non-optimal reasoning thoughts generated during the tree-search process, which naturally provides additional preference information. Specifically, ToT inherently generates multiple alternative thoughts at each reasoning step, and pruning is performed according to their evaluated qualities. This tree-search process constitutes a preference over all *intermediate* thought candidates—thoughts appearing in the best-discovered reasoning path are preferred over those that do not. Moreover, this could shed even more insights than the final best-discovered reasoning path, as non-optimal reasoning paths (and thus preferences) exist at each step in the tree-search.

Inspired by recently developed reinforcement learning from human feedback (RLHF) techniques like direct preference optimization (DPO) [13], we propose *Chain-of-Preference Optimization* (CPO) to fully exploit the inherent preference information. Specifically, we construct paired preference thoughts at each reasoning step according to the search tree of ToT and then train LLMs to align with these preferences using the DPO algorithm (as illustrated in Figure 1(c)). The paired preference thoughts are constructed based on the above intuition: at each reasoning step, we categorize thoughts as preferred or dispreferred based on their inclusion in the final paths chosen by ToT. With such preference data, CPO enables LLMs to generate the path preferred by ToT using CoT decoding at inference time.

We conduct extensive experiments to evaluate the effectiveness of CPO. Experiments on seven datasets using LLaMA [14] and Mistral [15] as base models demonstrate that CPO is highly effective in teaching LLMs the preferred thoughts of ToT at each reasoning step, leading to an average accuracy improvement of up to 4.3% compared to the base models. Additionally, the experiments reveal that CPO can achieve comparable or even superior performance to the ToT method, which on average requires more than 50 times longer for inference.

2   Related Work

Reasoning with LLMs.   LLMs have been shown to perform better when prompted to engage in multi-step reasoning [1, 2, 3]. Many studies have focused on improving the generated reasoning paths

by post-editing [16] or accessing external knowledge [3, 17]. A distinct approach, more relevant to our interests, transforms the linear reasoning structure into a non-linear format such as a tree or graph [18, 19, 20, 8, 9, 21], which combines thought evaluation with search algorithms like depth-first search (DFS) [22]. Different from our proposed CPO, these methods require searching during inference, which significantly increases latency.

LLM self-improving. Reinforcement learning (RL) has increasingly been applied to LLMs by treating them as RL agents for alignment with human feedback [23, 24, 25, 26]. Recent advances demonstrate the potential of using LLMs for self-generating data to augment fine-tuning processes [27, 28, 29, 30, 31, 32]. For instance, reinforced self-training methods [33, 34, 35, 36, 37] introduce mechanisms to curate new high-quality examples and iteratively enrich the training dataset for enhancing model performance. Nevertheless, these methods typically rely on either an external reward model [35, 34] or labeled data [33]. In contrast, approaches like self-rewarding [29, 38] utilize LLMs themselves to evaluate the generated content, aligning more closely with our method. However, these strategies still require initial seed data [29, 38], necessitating human annotation. Our work differs from previous methods as it does not rely on any ground-truth data, allowing LLMs to self-learn from their own feedback. Additionally, our approach constructs feedback in a chain fashion, focusing on reasoning steps, an aspect overlooked by prior works.

Monte Carlo tree-search for LLMs. Monte Carlo tree-search (MCTS) is a robust algorithm for navigating complex decision-making environments, commonly employed in strategic board games such as AlphaGo [39, 40, 41, 42, 43]. MCTS methodically constructs a search tree, balancing exploration and exploitation, simulates various outcomes, and updates utility estimates based on these simulations. Recent studies have shown that MCTS can enhance the decoding process in LLMs [11, 44, 45, 21, 12]. However, the primary challenge with MCTS is the high latency during inference, particularly in difficult reasoning tasks [46, 47]. While some approaches have attempted to optimize LLMs by leveraging reasoning paths identified through MCTS [11, 12], these methods still rely on labeled data and require separate policy and value models to explore and evaluate potential moves at the tree's leaves. In contrast, our CPO approach eliminates the need for human annotations and simplifies the tuning of LLMs without the necessity for additional models.

3 Background

In this section, we formalize our notation and provide a brief overview of key prior knowledge for our method. We denote language sequences by lowercase letters, e.g., x, y, z, to represent a sequence of tokens. The output distribution of an LLM parameterized by θ is denoted by π_θ.

3.1 Chain-of-Thought Prompting

Chain-of-thought (CoT) [1] is a method that prompts LLMs to generate a chain of reasoning steps before the final answer, as shown in Figure 1. It introduces a series of intermediate thoughts, denoted as $z_1, \cdots, z_n$, that link an input x to an output y, where n is the total number of reasoning steps. For instance, if x is a combination of demonstration examples and the input question and y is the final answer, each intermediate thought z_i forms a coherent language sequence representing a part of the overall reasoning path toward the final answer. The demonstration examples consist of a set of CoT demonstrations, which serve as exemplars in the prompting process. The intermediate reasoning thoughts are sequentially sampled from the distribution $z_i \sim \pi_\theta(\cdot|x, z_1, \cdots, z_{i-1})$ and the output is then derived from $y \sim \pi_\theta(\cdot|x, z_1, \cdots, z_n)$.

3.2 Tree-of-Thought Prompting

Tree-of-thought (ToT) [8] enables LLMs to explore multiple reasoning paths before answering a given question, as illustrated in Figure 1. This approach models the LLM reasoning task as a search over a tree, where each node represents a thought step in the reasoning path. ToT comprises two main components, both implemented through prompting LLMs: 1) the *thought generator* and 2) the *state evaluator*. The *thought generator* constructs several new thoughts for the next step based on the current state. Subsequently, the *state evaluator* generates scores for each new thought and selects the
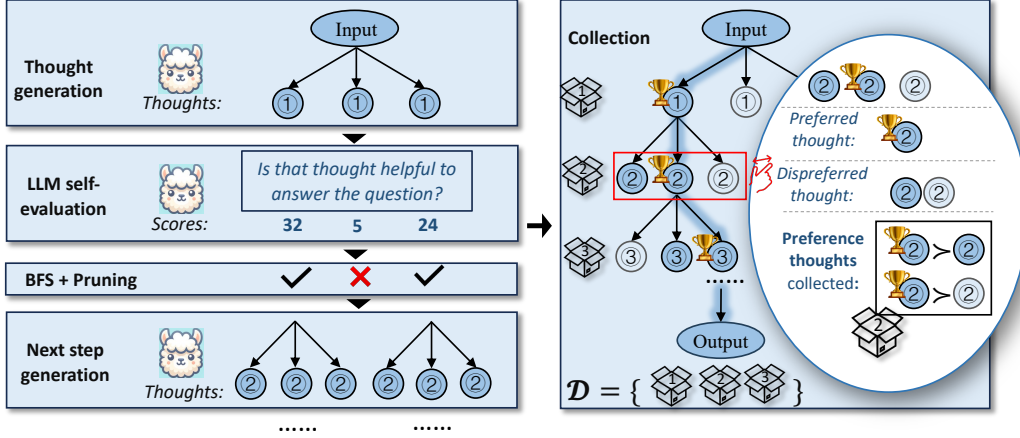
Figure 2: The framework of our CPO method. The left part illustrates the process of generating, evaluating, and pruning thoughts, while the right part demonstrates the collection of preference thoughts. The shaded path represents the final selected reasoning path. Thoughts marked with a trophy indicate preferred data, while sibling nodes without a trophy are marked as dispreferred.

n-best thoughts for further search. The final result is determined by the search algorithm (e.g., BFS or DFS) applied over the selected thoughts until the reasoning process reaches a conclusion.

3.3 Direct Preference Optimization

Direct preference optimization (DPO) is a method for directly optimizing an LLM to align with preference data [13], e.g., human feedback [13, 48, 49]. More specifically, RLHF traditionally frames the application of human feedback to enhance the performance of an LLM within the context of an RL problem. However, DPO reformulates the reward modeling and RL fine-tuning phases in RLHF to a single optimization problem. The objective function of DPO aims to maximize the ratio of probabilities for the preferred responses and optimize the LLM to imitate human preferences.

Given the generations $(\hat{y}_1, \hat{y}_2) \sim \pi(\hat{y}|x)$ conditioned on input x, these pairs are evaluated and ranked according to specific criteria. Preference data is then constructed from these ranked pairs, denoted by $\hat{y}_w \succ \hat{y}_l | x$, where $\hat{y}_w$ and $\hat{y}_l$ denote the preferred (winning) and dispreferred (losing) completions between $\hat{y}_1$ and $\hat{y}_2$, respectively. The DPO objective is formulated as follows:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\log \sigma \left(\beta \log \frac{\pi_\theta(\hat{y}_w|x)}{\pi_{\text{ref}}(\hat{y}_w|x)} - \beta \log \frac{\pi_\theta(\hat{y}_l|x)}{\pi_{\text{ref}}(\hat{y}_l|x)} \right), \tag{1}$$

where σ is the logistic function, the hyperparameter β regulates the penalty imposed for the deviations from the base reference model π_{ref}.

4 Our Method: Chain of Preference Optimization

Unlike previous methods that train LLMs to learn the complete reasoning paths [10, 50, 11, 12], our approach leverages the preferences over thoughts generated at each reasoning step, which are often discarded in prior works. Our key insight is that non-optimal thoughts generated during the tree-search process in ToT provide valuable preference information that can enhance LLM's reasoning ability. A major advantage of our method is that it utilizes this supervision only during training, thereby avoiding high inference latency. Our approach consists of two components: synthesizing the chain of preference thoughts (i.e., the preference thoughts in a chain fashion) and training with the CPO objective.

4.1 Synthesizing the Chain of Preference Thoughts

Our procedure for synthesizing and collecting preference thought pairs closely follows the inference process of ToT [8]. An overview of our method is shown in Figure 2. Specifically, the detailed process is divided into three parts: 1) *thought generation*, which generates multiple thoughts for each

reasoning step; 2) *state evaluation*, which evaluates each thought; and 3) *search and collection*, which finalizes the preference thoughts.

Thought generation.   Given a state $s_{i-1} = [x, z_1, \cdots, z_{i-1}]$ representing a partial solution with the input x and the sequence of thoughts $[z_1, \cdots, z_{i-1}]$ so far, we sample k thoughts for the next reasoning step:

$$z_i^{(j)} \sim \pi_\theta(z_i|s_{i-1}) = \pi_\theta(z_i|x, z_1, \cdots, z_{i-1}), \quad \text{for } j = 1, \cdots, k. \tag{2}$$

Conditioned on the initial input x, which contains the demonstration examples and the question to be answered, and the previous thoughts $z_1, z_2, \cdots, z_{i-1}$, the LLM generates multiple thoughts for the next reasoning step. Specifically, it follows the format of demonstrations, starting with the prefix "`Step` i," and samples k thoughts $\{z_i^{(j)}\}_{j=1}^k$. We control the model to pause at the end of $z_i^{(j)}$ by setting the generation of the string "`Step` $i+1$," as the stop criteria.[2] As a result, we obtain k new states $s_i^{(j)} = [x, z_1, \cdots, z_{i-1}, z_i^{(j)}]$ for $j = 1, \cdots, k$.

State evaluation.   Given different states $\{s_i^{(j)}\}_{j=1}^k$, we utilize the LLM to reason about the states and evaluate their progress toward solving the problem, eliminating the need for an external reward model or human annotations. To evaluate state $s_i^{(j)}$, the input to the LLM includes specific demonstration examples for the evaluation process, the input question x, and all the thoughts in the state (i.e., $[z_1, \cdots, z_{i-1}, z_i^{(j)}]$). The LLM follows the format of demonstrations to generate a verbal justification first, followed by a classification result from two classes: `likely` and `impossible`. The classification results are then used to assign a score, with `likely` $= 10$ and `impossible` $= 1$.

The prompt template used in our evaluation consists of two parts: (1) the general guidelines, and (2) task-specific demonstration examples. To minimize the effects of randomness and bias, we shuffle the order of demonstration examples [51] and repeatedly sample the generated justification and evaluation results. We then calculate the average score for the state $s_i^{(j)}$. The general guideline prompt for the evaluation is as follows: `Evaluate whether the thought helps in partially or directly answering the original question (likely/impossible).`

Search and collection.   We use BFS with pruning as the search algorithm to select the reasoning paths. After evaluation, we retain the n-best thoughts with the highest evaluation scores and proceed to the next step of generation. When the LLM generates a thought containing "`so the final answer is:`", the search algorithm concludes and returns the selected paths.

As shown in the right part of Figure 2, after finalizing the reasoning paths, the thoughts within the selected paths are marked as preferred (i.e., winning) thoughts. For each preferred thought at the i-th step z_i^w, we construct corresponding dispreferred (i.e., losing) thoughts. First, we identify the parent state s_{i-1}^w, which includes all the previous thoughts leading to z_i^w. Each child thought of s_{i-1}^w that is not included in the selected path is chosen as a dispreferred thought z_i^l compared to z_i^w. This process results in the preference pair (z_i^w, z_i^l) for the state s_{i-1}^w. We highlight that the constructed dataset $\mathcal{D}$ includes *preference data at every step of the reasoning chain*. This per-step paired preference supervision is usually overlooked in previous methods [11, 12].

4.2   Training with the CPO Objective

Once we have obtained the chain of preference thoughts $\mathcal{D}$, we can proceed with optimization. For the i-th step, given the previous reasoning thoughts s_{i-1}^w, the probabilities of generating z_i^w and z_i^l are denoted as $\pi_\theta(z_i^w|x, s_{i-1}^w)$ and $\pi_\theta(z_i^l|x, s_{i-1}^w)$, respectively. To optimize the LLM on this pair of preference thoughts, we can directly substitute it into Equation 1:

$$\mathcal{L}_i(\pi_\theta; \pi_{\text{ref}}) = -\log \sigma \left(\beta \log \frac{\pi_\theta(z_i^w|x, s_{i-1}^w)}{\pi_{\text{ref}}(z_i^w|x, s_{i-1}^w)} - \beta \log \frac{\pi_\theta(z_i^l|x, s_{i-1}^w)}{\pi_{\text{ref}}(z_i^l|x, s_{i-1}^w)} \right). \tag{3}$$

---

[2]The "stop criteria" is used to control when generation should stop, which is implemented via a function input in Hugging Face's Transformers Library.

Thus, the objective function for CPO can be formulated as follows:

$$\mathcal{L}_{\text{CPO}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{(x, z_i^w, z_i^l, s_{i-1}^w) \sim \mathcal{D}} \left[\mathcal{L}_i(\pi_\theta; \pi_{\text{ref}}) \right]. \tag{4}$$

5 Experiments

In this section, we empirically validate that CPO improves the reasoning ability of the base model, and uncover several insightful findings.

5.1 Settings

Datasets and evaluation metrics. We focus our research on three types of reasoning tasks: *Question Answering* (QA), *Fact Verification*, and *Arithmetic Reasoning*. For QA, we conduct experiments on three widely used datasets: Bamboogle [17], WikiMultiHopQA [52], and HotpotQA [53]. For fact verification, we use three datasets: Fever [54], Feverous [55], and Vitaminc [56]. For arithmetic reasoning, we test on the SVAMP dataset [57]. Our choice of tasks was driven by the performance of the models using the ToT method, which showed improvements in QA, fact verification, and arithmetic reasoning. We use 4-shot prompting for each dataset, with CoT demonstrations manually constructed by previous works [1, 17, 2]. Detailed experimental configurations can be found in Appendix A. For evaluation metrics, we report the accuracy and the average latency of generating the answer per instance. More metrics can be found in Appendix B.

Baselines. To validate the effectiveness of our proposed CPO, we consider the following baselines: 1) CoT [1], which prompts the LLM to generate a series of reasoning steps before producing the final answer. In our experiments, we use CoT with greedy decoding to assess the model's reasoning capabilities without any tuning. 2) ToT [8], which requires the LLM to explore multiple reasoning paths via tree search before generating the final answer. We use ToT to select reasoning paths and construct datasets to improve LLM's reasoning ability in the following TS-SFT baseline and our CPO method. 3) TS-SFT [11], which finds reasoning paths through tree search (i.e., ToT in our implementation) and then uses these paths during the supervised fine-tuning (SFT) process (referred to as SFT in Section 5.3 and 6).

Implementation Details Our experiments are based on widely used LLMs, specifically LLaMA2-7B/13B [14] and Mistral-7B [15]. For efficient fine-tuning, we use Low-Rank Adaptation (LoRA) adapters [58]. In all experiments, we set the regularization controller β to 0.1, generate 10 new thoughts for each state, and retain the top 5 thoughts after pruning at each step of reasoning. The temperature is set to 0.9 for SVAMP and 0.4 for the other datasets. The learning rates for DPO and SFT are 5e-6 and 1e-5, respectively. We use a batch size (with accumulation) of 32 and optimize the LLM with AdamW [59]. For LoRA, the rank is set to 8, and α is set to 16. All experiments are conducted on NVIDIA A100 GPUs. The latency reported in Table 1 is based on a single NVIDIA A100 40GB. Both training and inference are performed using the Accelerate [60] backend. We train the LLMs for 4 epochs with early stopping based on the performance on a randomly sampled validation set. To mitigate the influence of randomness, all experiments are repeated three times with different random seeds, and the average results are reported.

5.2 Overall Results on Reasoning

Table 1 summarizes the performance across various reasoning tasks. We have the following findings:

CPO improves LLM's reasoning ability. As shown in Table 1, CPO enhances the reasoning ability of the base LLM, achieving an average improvement of 4.3% and a maximum improvement of 9.7% across all tasks and LLMs compared to the CoT approach. This indicates that CPO effectively improves the LLM's reasoning capabilities. Notably, CPO achieves these improvements without requiring additional human-annotated data, which is particularly beneficial in resource-constrained settings.

CPO has a lower latency than ToT but comparable performance. Although ToT consistently improves performance over CoT, it incurs high latency due to the need to generate and evaluate multiple thoughts at each reasoning step during inference. This process produces numerous tokens, resulting

Table 1: Experimental results for ToT, CoT, TS-SFT, and our proposed CPO across complex tasks including question answering, fact verification, and arithmetic reasoning are presented. * means significantly better than the best baseline (TS-SFT) with $p < 0.01$. **Bold** denotes the best method, and the second best if the top method is ToT.

		ToT [8]		CoT [1]		TS-SFT [11]		CPO (ours)	
		Acc. (%)↑	Latency (s/ins.)↓	Acc. (%)↑	Latency (s/ins.)↓	Acc. (%)↑	Latency (s/ins.)↓	Acc. (%)↑	Latency (s/ins.)↓
LLaMA2-7B									
Question Answering	Bam.	**33.6**	1168.4	29.6	37.2	30.4	36.5	**32.0***	38.2
	2Wiki.	28.6	847.6	26.3	35.7	27.6	35.5	**29.7***	35.7
	Hot.	23.0	1100.7	21.0	45.5	22.7	44.8	**24.0***	41.1
Fact Verification	FVR.	47.3	2087.1	45.8	33.8	47.5	34.0	**53.2***	36.8
	FVRS.	47.5	2539.5	44.3	40.6	46.0	40.4	**49.0***	41.2
	Vita.	50.7	2639.3	47.3	35.9	51.0	40.1	**52.7***	40.1
Arithmetic	SVA.	42.7	1861.1	37.7	33.3	43.1	30.2	**46.0***	32.1
Average Performance		39.1	1749.1	36.0	37.4	38.3	37.4	**40.9***	37.9
LLaMA2-13B									
Question Answering	Bam.	**53.8**	1318.3	48.0	50.5	50.8	49.6	**52.0***	50.3
	2Wiki.	**36.3**	1097.1	28.3	67.0	29.0	66.5	**30.3***	60.4
	Hot.	**32.0**	1271.0	29.0	65.2	**30.3**	65.5	**30.3**	63.8
Fact Verification	FVR.	48.8	3139.8	48.2	45.4	48.8	44.0	**49.2**	43.9
	FVRS.	**51.3**	3433.2	50.0	61.1	48.8	58.0	**50.7***	68.2
	Vita.	52.5	2933.6	46.3	48.3	49.7	51.5	**54.0***	58.6
Arithmetic	SVA.	45.7	2115.3	40.3	46.2	44.6	46.4	**50.0***	48.1
Average Performance		**45.8**	2186.9	41.4	54.8	43.1	54.5	**45.2***	56.2
Mistral-7B									
Question Answering	Bam.	**46.4**	4399.6	41.6	46.2	41.6	45.0	**45.6***	47.0
	2Wiki.	28.4	2356.9	26.7	45.1	31.0	44.2	**31.7**	46.3
	Hot.	**30.0**	4698.0	28.0	58.4	28.6	56.2	**29.4**	56.9
Fact Verification	FVR.	**61.4**	3291.3	57.9	41.8	**60.2**	41.7	59.9	40.6
	FVRS.	50.5	5537.8	48.0	51.0	49.7	49.5	**54.0***	53.0
	Vita.	52.2	4698.2	47.7	44.8	50.3	45.9	**53.7***	46.6
Arithmetic	SVA.	66.0	4623.7	65.3	41.4	59.0	41.3	**69.3***	44.9
Average Performance		47.8	4229.4	45.0	47.0	45.8	46.3	**49.1***	47.9

in significant computational and memory overhead [61]. In contrast, CPO shifts this computational burden to the training phase, maintaining the low latency of CoT (i.e., $57.5\times$ faster than ToT on average) during inference while providing comparable or superior performance. This demonstrates that our CPO can deliver enhanced reasoning capabilities without compromising efficiency.

CPO surpasses TS-SFT on average. Despite both CPO and TS-SFT using ToT to generate training data (where our implementation of ToT remains consistent), CPO exhibits an average improvement of 2.7% and reaches a maximum increase of 10.3%. A key factor behind this performance is the CPO's ability to fully utilize ToT's reasoning process. Specifically, CPO effectively leverages both selected and unselected thoughts at each reasoning step, whereas TS-SFT only uses information from the selected paths, offering CPO with a clear advantage. A detailed discussion of the effectiveness of CPO is presented in Section 5.3.

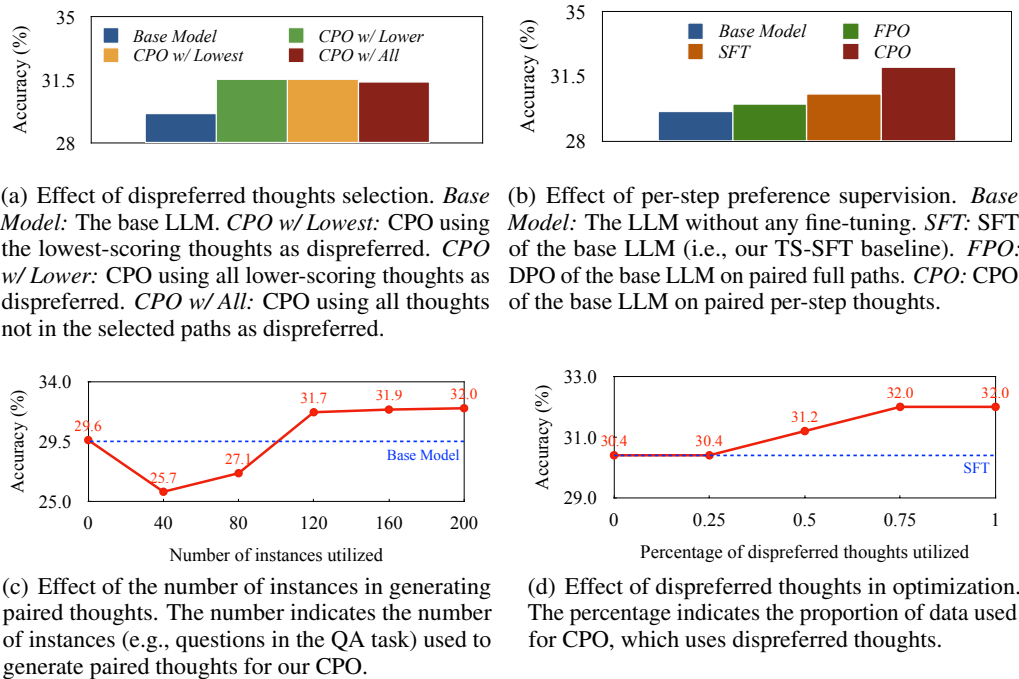## 5.3 Component-wise Evaluations

(a) Effect of dispreferred thoughts selection. *Base Model:* The base LLM. *CPO w/ Lowest:* CPO using the lowest-scoring thoughts as dispreferred. *CPO w/ Lower:* CPO using all lower-scoring thoughts as dispreferred. *CPO w/ All:* CPO using all thoughts not in the selected paths as dispreferred.

(b) Effect of per-step preference supervision. *Base Model:* The LLM without any fine-tuning. *SFT:* SFT of the base LLM (i.e., our TS-SFT baseline). *FPO:* DPO of the base LLM on paired full paths. *CPO:* CPO of the base LLM on paired per-step thoughts.

(c) Effect of the number of instances in generating paired thoughts. The number indicates the number of instances (e.g., questions in the QA task) used to generate paired thoughts for our CPO.

(d) Effect of dispreferred thoughts in optimization. The percentage indicates the proportion of data used for CPO, which uses dispreferred thoughts.

Figure 3: Component-wise evaluations and analysis on the Bamboogle dataset using the LLaMA2-7B as the base model.

Effect of selection methods of dispreferred thoughts. We analyze the impact of different methods for selecting dispreferred thoughts on model performance. As shown in Figure 4, we experiment with three strategies based on evaluation scores for each thought: 1) *CPO w/ Lowest*: Only the lowest-scoring thoughts in each reasoning step are dispreferred thoughts. 2) *CPO w/ Lower*: Thoughts with evaluation scores lower than the selected paths are dispreferred thoughts. 3) *CPO w/ All*: All thoughts not in the selected paths are considered dispreferred thoughts. We ensured an equal number of training samples for each strategy. Note that the evaluation score at each intermediate reasoning step (apart from the final one) determines whether to create the next reasoning step but not which thoughts are preferred. For example, as shown in the figure, even though the score of 32 is higher than 23, the thought with a score of 23 is preferred since it is part of the selected path.
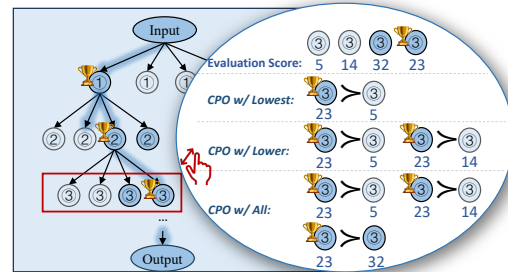

Figure 4: Different strategies for selecting dispreferred thoughts and their impact on model performance. At each reasoning step, three strategies are used to select dispreferred thoughts based on their reasoning scores: 1) *CPO w/ Lowest*: Selects only the thought with the lowest score. 2) *CPO w/ Lower*: Selects all thoughts with scores lower than the preferred thought. 3) *CPO w/ All*: Selects all thoughts as dispreferred as long as they are not the preferred thought.

The results in Figure 3(a) show that the performance differences among these strategies are minimal. This suggests that the distinction between preferred and dispreferred thoughts is better determined in the selected reasoning path rather than intermediate evaluation scores. To obtain a greater number of preferred thoughts for each instance to create paired preference thoughts, we chose the *CPO w/ All* strategy.

Effect of numbers of training data. To assess the impact of the number of training data used in optimization, we conduct an ablation analysis by varying the number of instances (e.g., questions in the QA task) used to generate paired preference thoughts, ranging from 0 to 200. As illustrated in

8

Table 2: Effect of different kinds of training data on the Bamboogle dataset using the LLaMA2-7B as the base model.

Data	Description	SFT	CPO
Single-Task	Training only on specific task (Bamboogle) data.	30.4	**32.0**
Uniform QA	Training on 3 datasets of the same type (QA) as the test task.	31.2	**35.2**
Mixed-Type	Training on all 7 different types of data.	29.6	**35.2**

Figure 3(c), we observe that with an increase in the number of instances, the model's performance initially declines and then rises. Specifically, when trained with data generated from less than 80 instances, the model's performance is even worse than without any training, likely due to overfitting [62], which leads to performance degradation. However, as the number increases to 120, the model's performance consistently improves. Optimizing with paired thoughts from 120 instances, the model's performance surpasses that of the base model. When the number exceeds 120, the model's performance converges, indicating a balance of data for training.

Sensitivity of data mixture. We explore the performance of the CPO method across diverse data settings to assess its adaptability and learning efficiency from various data types. As shown in Table 2, we specifically examine three different data configurations: 1) single task data, 2) uniform QA data, and 3) mixed-type data. Our findings indicate that CPO demonstrates performance improvements of 3.2% in both settings 2 and 3, suggesting its robust ability to harness diverse data sources to enhance learning outcomes. In contrast, the SFT method exhibits comparable performance across these settings, indicating a different sensitivity to data diversity. It is worth noting that, to ensure fairness, although we find that mixed data leads to better performance, the experiments in Table 1 are conducted using individual datasets for training, consistent with the baselines.

6 Analysis

Do we need dispreferred information? We explore the impact of dispreferred thoughts on model performance by gradually incorporating these thoughts into the training data. Initially, we introduce dispreferred thoughts for their corresponding preferred counterparts and apply CPO to this segment of the data. For preferred thoughts without dispreferred counterparts, we implement SFT on these data. Consequently, the percentage of dispreferred thoughts incorporated can also be viewed as the proportion of data processed using CPO. We adjust the inclusion percentage of dispreferred thoughts from 0% to 100%. An inclusion of 0% indicates that we utilize SFT solely on the preferred thoughts, i.e., the baseline TS-SFT. Conversely, an inclusion of 100% signifies our CPO, where the entire dataset includes paired preferred and dispreferred thoughts.

Why is chain level optimization important?
As shown in Figure 3(d), we find that increasing the percentage of dispreferred data inclusion consistently improves model performance. This suggests that dispreferred thoughts are beneficial during the optimization process, highlighting the importance of leveraging both preferred and dispreferred thoughts for enhancing the model's reasoning capabilities.

Unlike our CPO, an alternative approach is to construct preference data using complete reasoning paths, i.e., using the selected full reasoning paths as preferred and other paths as dispreferred data, as shown in Figure 5. This method essentially applies DPO at the full-path level, referred to here as Full-path Preference Optimization (FPO). However, FPO encounters a significant issue where the gradients of the longest common prefix (LCP) tokens in paired data cancel
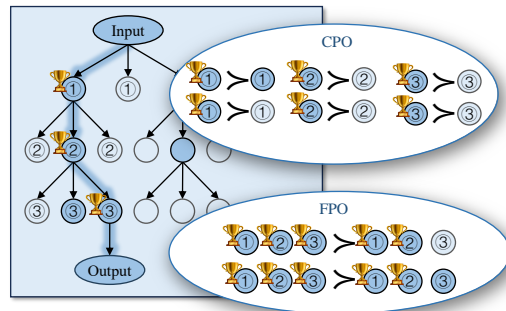

Figure 5: Illustrations of two different ways to construct paired preference data: 1) CPO: Paired preference data are constructed at each thought step. 2) FPO: Paired preference data are constructed only at the full path level.

out, which we call the *LCP gradient cancellation* issue. For example, for the preferred path $\hat{y}_w = [5, +, 4, =, 9, and, 9, +, 2, =, 11]$ and the dispreferred path $\hat{y}_l = [5, +, 4, =, 9, and, 9, +, 2, =, 15]$, the gradient will only be computed for the last token where the two sequences diverge.

To mathematically illustrate how LCP gradient cancellation happens in FPO, consider $\hat{y}_w = [p_{1:n}, w_{n+1}]$ and $\hat{y}_l = [p_{1:n}, l_{n+1}]$, where p is the longest common prefix sequence between $\hat{y}_w$ and $\hat{y}_l$. The gradient of FPO is given by:

$$\nabla_\theta \mathcal{L}_{\text{FPO}}(\pi_\theta; \pi_{\text{ref}}) = C(\theta) \cdot \nabla_\theta \left(\log \pi_\theta(\hat{y}_w|x) - \log \pi_\theta(\hat{y}_l|x) \right)$$

$$= C(\theta) \cdot \nabla_\theta \left(\boxed{\log \pi_\theta(p_{1:n}|x)} + \log \pi_\theta(w_{n+1}|x, p_{1:n}) - \boxed{\log \pi_\theta(p_{1:n}|x)} - \log \pi_\theta(l_{n+1}|x, p_{1:n}) \right),$$

where $C(\theta)$ is a scalar function that does not affect the direction of the gradient and can be absorbed into the learning rate.

We can clearly see that the gradient terms of the common prefix tokens (highlighted with boxes) cancel each other out. This issue also exists in DPO training [63], but FPO suffers more frequently and severely due to the longer LCP between paired data constructed by tree search. As an empirical evidence, we observe the LCP length accounts for 28% of the total length in the Bamboogle dataset. CPO, on the other hand, constructs preference data at every step in the reasoning chain, allowing optimization of the LLM on all steps in the reasoning path. This means the common prefix can be optimized at its own step, ensuring that the gradient still exists for the common prefix.

We also compare FPO to CPO empirically in Figure 3(b), which further substantiates this observation. Switching to FPO led to a relative performance decrease of 4.6%, even worse than the baseline SFT that does not utilize any information from dispreferred data. This underscores the importance of per-step preference thoughts for CPO.

7 Conclusion

In this work, we introduce a novel method called Chain of Preference Optimization (CPO), which leverages the supervision generated by the self-reasoning process (i.e., tree-of-thoughts) to enhance the reasoning ability of LLMs. Experiments on three different LLMs across seven different datasets demonstrate that CPO can consistently improve the performance of the base model by 4.3% on average without sacrificing inference speed. Furthermore, our method also substantially outperforms the strong baseline TS-SFT and even achieves comparable performance to the ToT method, which requires approximately 57.5 times more inference time.

In future work, we plan to integrate CPO with other reasoning algorithms, such as Graph-of-Thoughts [9] and AlphaZero-like tree search [11]. Furthermore, we intend to explore the potential of using a weaker LLM to evaluate a stronger one within the CPO framework, facilitating weak-to-strong alignment [64].

Acknowledgement

References

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

[2] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2022.

[3] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*, 2022.

[4] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[5] Ting Zhang, Ivana Clairine Irsan, Ferdian Thung, and David Lo. Cupid: Leveraging chatgpt for more accurate duplicate bug report detection. *arXiv preprint arXiv:2308.10022*, 2023.

[6] Xinyang Hu, Fengzhuo Zhang, Siyu Chen, and Zhuoran Yang. Unveiling the statistical foundations of chain-of-thought prompting methods. *arXiv preprint arXiv:2408.14511*, 2024.

[7] Fengzhuo Zhang, Boyi Liu, Kaixin Wang, Vincent Tan, Zhuoran Yang, and Zhaoran Wang. Relational reasoning via set transformers: Provable efficiency and applications to marl. *Advances in Neural Information Processing Systems*, 35:35825–35838, 2022.

[8] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[10] Fangkai Jiao, Chengwei Qin, Zhengyuan Liu, Nancy F Chen, and Shafiq Joty. Learning planning-based reasoning by trajectories collection and process reward synthesizing. *arXiv preprint arXiv:2402.00658*, 2024.

[11] Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

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

[13] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[15] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[16] Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *ACL*, pages 5823–5840, Toronto, Canada, July 2023.

[17] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

[18] Fangkai Jiao, Yangyang Guo, Xuemeng Song, and Liqiang Nie. MERIt: Meta-path guided contrastive learning for logical reasoning. In *Findings of ACL*. ACL, 2022.

[19] Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*, 2023.

[20] Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

[21] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NeurIPs*, 35:27730–27744, 2022.

[24] Jing Xu, Megan Ung, Mojtaba Komeili, Kushal Arora, Y-Lan Boureau, and Jason Weston. Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback. *arXiv preprint arXiv:2208.03270*, 2022.

[25] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[26] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*, 2023.

[27] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[28] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

[29] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

[30] Fangkai Jiao, Zhiyang Teng, Bosheng Ding, Zhengyuan Liu, Nancy F. Chen, and Shafiq R. Joty. Exploring self-supervised logic-enhanced training for large language models. In *NAACL*. Association for Computational Linguistics, 2024.

[31] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 2024.

[32] Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Oleksandr Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. Learning math reasoning from self-sampled correct and partially-correct solutions. *arXiv preprint arXiv:2205.14318*, 2022.

[33] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[34] Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, et al. Rest meets react: Self-improvement for multi-step reasoning llm agent. *arXiv preprint arXiv:2312.10003*, 2023.

[35] Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. React meets actre: When language agents enjoy training data autonomy, 2024.

[36] Chaojie Wang, Yanchen Deng, Zhiyi Lv, Shuicheng Yan, and An Bo. Q*: Improving multi-step reasoning for llms with deliberative planning. *arXiv preprint arXiv:2406.14283*, 2024.

[37] Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. Bootstrapping language models with dpo implicit rewards. *arXiv preprint arXiv:2406.09760*, 2024.

[38] Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipali, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. *arXiv preprint arXiv:2403.15042*, 2024.

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

[40] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[41] Mark HM Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. Monte-carlo tree search solver. In *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6*, pages 25–36. Springer, 2008.

[42] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2008.

[43] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[44] Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Making ppo even better: Value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*, 2023.

[45] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Everything of thoughts: Defying the law of penrose triangle for thought generation. *arXiv preprint arXiv:2311.04254*, 2023.

[46] Zichen Liu, Siyi Li, Wee Sun Lee, Shuicheng Yan, and Zhongwen Xu. Efficient offline policy optimization with a learned model. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.05980.

[47] Jessica B Hamrick, Abram L Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. On the role of planning in model-based deep reinforcement learning. *arXiv preprint arXiv:2011.04021*, 2020.

[48] Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024.

[49] Zixuan Liu, Xiaolin Sun, and Zizhan Zheng. Enhancing llm safety via constrained direct preference optimization. *arXiv preprint arXiv:2403.02475*, 2024.

[50] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR, abs/2312.08935*, 2023.

[51] Xavier Amatriain. Prompt design and engineering: Introduction and advanced methods. *arXiv preprint arXiv:2401.14423*, 2024.

[52] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

[53] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[54] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, 2018.

[55] Rami Aly, Zhijiang Guo, Michael Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. Feverous: Fact extraction and verification over unstructured and structured information. *arXiv preprint arXiv:2106.05707*, 2021.

[56] Tal Schuster, Adam Fisch, and Regina Barzilay. Get your vitamin c! robust fact verification with contrastive evidence. *arXiv preprint arXiv:2103.08541*, 2021.

[57] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

[58] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[60] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

[61] Jinwei Yao, Kaiqi Chen, Kexun Zhang, Jiaxuan You, Binhang Yuan, Zeke Wang, and Tao Lin. Deft: Flash tree-attention with io-awareness for efficient tree-search-based llm inference. *arXiv preprint arXiv:2404.00242*, 2024.

[62] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR, 2024.

[63] Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*, 2024.

[64] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschen-brenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

[65] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

[66] Cunxiao Du, Zhaopeng Tu, and Jing Jiang. Order-agnostic cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 2849–2859. PMLR, 2021.

[67] Cunxiao Du, Zhaopeng Tu, Longyue Wang, and Jing Jiang. ngram-OAXE: Phrase-based order-agnostic cross entropy for non-autoregressive machine translation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5035–5045, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[68] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[69] Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, et al. Glide with a cape: A low-hassle method to accelerate speculative decoding. In *Forty-first International Conference on Machine Learning*, 2024.

[70] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[71] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[72] Cunxiao Du, Hao Zhou, Zhaopeng Tu, and Jing Jiang. Revisiting the markov property for machine translation. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 582–588, 2024.

[73] Cunxiao Du, Zhaozheng Chen, Fuli Feng, Lei Zhu, Tian Gan, and Liqiang Nie. Explicit interaction model towards text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6359–6366, 2019.

[74] Xuan Zhang and Wei Gao. Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method. *arXiv preprint arXiv:2310.00305*, 2023.

[75] Xuan Zhang and Wei Gao. Reinforcement retrieval leveraging fine-grained feedback for fact checking news claims with black-box llm. *arXiv preprint arXiv:2404.17283*, 2024.

[76] Wenxiang Jiao, Wenxuan Wang, J-t Huang, Xing Wang, and Zhaopeng Tu. Is chatgpt a good translator? a preliminary study. arxiv. *arXiv preprint arXiv:2301.08745*, 2023.

[77] Ting Zhang, Ivana Clairine Irsan, Ferdian Thung, and David Lo. Revisiting sentiment analysis for software engineering in the era of large language models. *arXiv preprint arXiv:2310.11113*, 2023.

[78] Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv e-prints*, pages arXiv–2404, 2024.

Societal Impacts and Limitations

Since our CPO does not require any human annotation, it can be directly used. For example, to protect the safety of large models, one can simply provide a constitution, and then fine-tune the LLM to make it more compliant. This also introduces another issue: our method can be adjusted for malicious applications. Our limitation is that we still need to generate data through ToT, which is a time-consuming process. We aim to accelerate the complex reasoning processes during training data generation by incorporating methods like non-autoregressive generation [65, 66, 67], speculative decoding [68, 69], and KV cache pruning [70, 71, 72]. Additionally, we have only tested this on text language models and have not tried it on vision-language models. Furthermore, the application scope of our method remains restricted to a small set of downstream tasks. Expanding its application to diverse tasks, such as text classification [73], news verification [74, 75], machine translation [76], and software engineering [77], remains an area for future research. Moreover, ethical considerations must be taken into account, as the potential for misuse could lead to unintended consequences.

A   Detailed Experiment Configurations

To maintain a reasonable budget, especially given the high computational demand of ToT, we limit each dataset to a maximum of 300 test samples through random sampling. For datasets that contain less than 300 test samples, we instead use all available samples. For training, we randomly select less than 300 instances from each dataset to construct the preference data pairs, **without** using the ground-truth labels. This is because we observe that more number of training data does not lead to performance improvement as shown in Section 5.3. In addition, in our experiments, approximately 200 samples (e.g., questions in the QA task) on average could generate about 6,531 preference pairs, suggesting that our CPO requires only a small number of samples by design. Constructing preference data is a time-intensive process. The choice of 300 samples represents a practical trade-off between efficiency and effectiveness, allowing us to manage resources effectively while still achieving noticeable improvements.

B   Additional Experiments

CPO benefits from iterative learning.   Inspired by the iterative improvements achieved in previous research [28, 78], in this section, we explore whether CPO can be further improved by iterative learning. Specifically, we try two distinct iterative training strategies: *1. SFT+CPO*: in *iter=0*, Start with a base LLM that has not been fine-tuned at all; in *iter=1*, SFT the base LLM on the reasoning path selected by ToT (base model); in *subsequent iterations (iteration >1)*, Continue to fine-tune the model using the CPO method, based on the chain of preference thoughts constructed by the model in the previous iterations. and *2. CPO only*: in *iter=0*, same as *iter=0* in *SFT+CPO*; in *subsequent iterations (iteration >0)*: Only use the CPO method for training in all iterations, similar to the approach in *SFT+CPO* after the first iteration.

As shown in Table 3, We find that if use CoT for inference, as the number of iterations increases, the performance of the model gradually improves. In the *CPO only* setting, the performance improves by 4% after two iterations. However, an intriguing phenomenon is noted: if we use the ToT method for inference on our fine-tuned models, the performance does not consistently rise and sometimes even declines. For instance, in the *SFT+CPO* setting, after the first round of SFT, the performance with ToT decreased by 2.7%. We hypothesize this may be related to a decrease in the diversity of the model's outputs after fine-tuning, which reduces the search space for ToT, making it challenging to find better reasoning paths. When the performance of CoT and ToT becomes similar, further fine-tuning of the LLM leads to convergence in the *SFT+CPO* setting and even a decline in the *CPO only* setting.

Comparison with ReST and self-rewarding baseline.   The settings of ReST [33] and self-rewarding [28] are different from ours, as discussed in Section 2. These methods rely on either external reward models (ReST) or labeled data (self-rewarding), which makes them not directly comparable to our approach. To ensure a fair comparison with our CPO method, we prompted the LLM itself in the same way as our CPO to serve as the reward model for ReST and as the labeled data

Inference Method	SFT+CPO				CPO only			
	iter=0 (Base)	iter=1 (SFT)	iter=2 (CPO)	iter=3 (CPO)	iter=0 (Base)	iter=1 (CPO)	iter=2 (CPO)	iter=3 (CPO)
CoT	29.6	30.4	30.4	31.2	29.6	32.0	33.6	31.2
ToT	33.6	30.4	31.2	-	33.6	34.4	33.6	-

Table 3: Effects of iterative learning on the Bamboogle dataset using the LLaMA2-7B as the base model.

annotator for self-rewarding, respectively. As shown in Table 4, the results indicate that, on average, our CPO method surpasses both ReST and self-rewarding under this fair comparison setting.

Model	Bam.	2wiki	Hotpotqa	Fever	Feverous	Vitaminc	Svamp	Average
LLaMA2-7b								
Rest	30.4	24.0	22.3	45.5	43.9	51.7	42.3	37.2
SelfR	31.2	25.3	21.0	48.8	44.7	51.3	43.0	37.5
CPO	32.0	29.7	24.0	53.2	49.0	52.7	46.0	40.9
LLaMA2-13b								
Rest	48.0	28.3	28.7	46.8	48.0	50.2	44.3	42.0
SelfR	48.0	29.0	30.0	47.8	48.5	51.0	45.3	42.8
CPO	52.0	30.3	30.3	49.2	50.7	54.0	50.0	45.2
Mistral-7b								
Rest	43.2	26.7	27.4	59.5	48.3	49.7	63.3	45.4
SelfR	44.0	28.0	28.1	58.2	48.0	50.0	65.0	45.9
CPO	45.6	31.7	29.4	59.9	54.0	53.7	69.3	49.1

Table 4: Performance comparison across different datasets.

More metrics on QA dataset. We include F1 scores for the three QA tasks in Table 5. The results show that the F1 performance aligns well with the corresponding accuracy for each task.

Ablation studies across datasets and models. Figure 6, 7, and 8 provide ablations and analysis across various models and datasets. The observed trends remain generally consistent across these different settings.
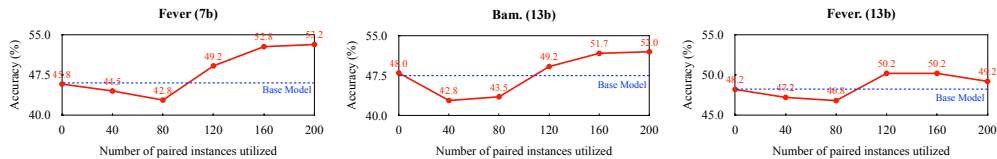

Figure 6: Effect of the number of instances in generating paired thoughts.

Illustrative examples of the reasoning paths preferred by CPO. Table 6 presents examples demonstrating that the paths favored by CPO align more closely with those selected by ToT than by CoT, indicating a higher reasoning quality in CPO's chosen paths.

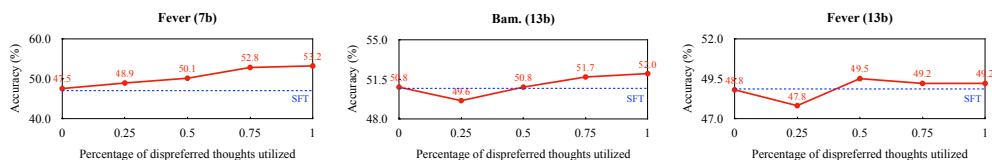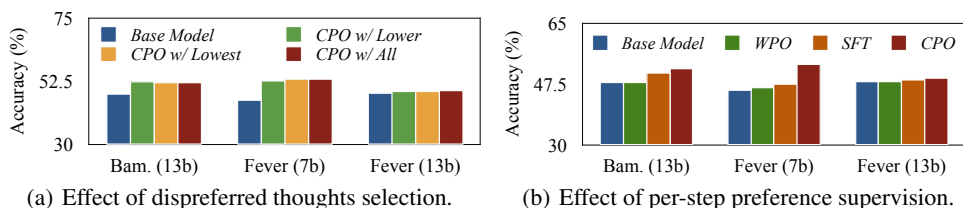|	LLaMA2-7b				LLaMA2-13b				Mistral-7b			
	ToT	CoT	SFT	CPO	ToT	CoT	SFT	CPO	ToT	CoT	SFT	CPO
Bam.	29.5	28.1	29.3	31.8	43.0	40.9	41.4	43.0	48.1	42.2	41.9	47.5
2wiki.	26.5	23.6	25.0	26.0	30.7	26.8	27.2	28.3	25.5	24.3	27.3	27.0
hot.	23.4	20.5	22.9	24.7	33.3	31.3	31.3	32.0	29.9	25.9	26.8	28.0

Table 5: F1 scores on QA datasets.

Figure 7: Effect of dispreferred thoughts in optimization.

(a) Effect of dispreferred thoughts selection.

(b) Effect of per-step preference supervision.

Figure 8: Component-wise Evaluations and Analysis.

Question 1: When did the author of Annabel Lee enlist in the army?
CoT: Step 1, when did Edgar Allan Poe enlist in the army? Edgar Allan Poe enlisted in the army on May 27, 1827. Step 2, when did Edgar Allan Poe die? Edgar Allan Poe died on October 7, 1849. Step 3, so the final answer is: October 7, 1849.
ToT: step 1, when did edgar allan poe enlist in the army? edgar allan poe enlisted in the army on august 16, 1827. step 2, what year did edgar allan poe enlist in the army? edgar allan poe enlisted in the army in 1827. step 3, so the final answer is: 1827.
CPO: step 1, when did Edgar Allan Poe die? Edgar Allan Poe died on October 7, 1849. Step 2, when did Edgar Allan Poe enlist in the army? Edgar Allan Poe enlisted in the army on May 27, 1827. Step 3, so the final answer is: May 27, 1827.
Question 2: What is the nickname of the easternmost U.S. state?
CoT: Step 1, who is the scientist at MIT that won the Queen Elizabeth Prize for Engineering in 2013? The scientist at MIT that won the Queen Elizabeth Prize for Engineering in 2013 was Daniela Rus. Step 2, who is the father of Daniela Rus? Daniela Rus's father is Vladimir Rus. Step 3, so the final answer is: Vladimir Rus.
ToT: step 1, what is the name of the easternmost u.s. state? maine is the easternmost u.s. state. step 2, what is the nickname of maine? maine's nickname is 'the pine tree state.
CPO: Step 1, what is the name of the easternmost U.S. state? The easternmost U.S. state is Maine. Step 2, what is the nickname of Maine? The nickname of Maine is The Pine Tree State. Step 3, so the final answer is: The Pine Tree State.

Table 6: Illustrative examples of the reasoning paths preferred by CPO. Similar paths between ToT and CPO are in yellow.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction of the paper clearly outline the main contributions and the scope of the research.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper adequately discusses the limitations of the work in Section 7.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: Theorems, formulas, and Lemmas in the paper are numbered and properly referenced.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: The paper provides a clear and comprehensive explanation of the proposed method. Key hyper-parameters critical for reproducing the results are meticulously detailed.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The code is provided in the supplemental material.

Guidelines:

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

6. **Experimental Setting/Details**

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

Answer: [Yes]

Justification: The experiment setting and details is presented in Section 5.1 and A.

Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [Yes]

Justification: The results are accompanied by significant tests.

Guidelines:

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

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: The information on the computer resources can be found in Section 5.1 and A.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research conform the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: The paper adequately discusses the impacts of the work in Section 7.

    Guidelines:

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

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

11. **Safeguards**

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

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The assets used in this paper are properly credited, and the license and terms are respected.

Guidelines:

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

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

13. **New Assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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