Reasoning Paths Optimization:
Learning to Reason and Explore From Diverse Paths

Yew Ken Chia[*][1,📧]   **Guizhen Chen**[*][1,2]   **Weiwen Xu**[†][1]

Luu Anh Tuan[2]   **Soujanya Poria**[📧]   **Lidong Bing**[1]

[📧] Singapore University of Technology and Design
[1]DAMO Academy, Alibaba Group, Singapore
[2]Nanyang Technological University, Singapore

{yewken_chia, sporia}@sutd.edu.sg   {guizhen001, anhtuan.luu}@ntu.edu.sg
{yewken.chia, guizhen.chen, xuweiwen.xww, l.bing}@alibaba-inc.com

Abstract

Advanced models such as OpenAI o1 exhibit impressive problem-solving capabilities through step-by-step reasoning. However, they may still falter on more complex problems, making errors that disrupt their reasoning paths. We attribute this to the expansive solution space, where each step has the risk of diverging into mistakes. To enhance language model reasoning, we introduce a specialized training framework called Reasoning Paths Optimization (RPO), which enables learning to reason and explore from diverse paths. Our approach encourages favorable branches at each reasoning step while penalizing unfavorable ones, enhancing the model's overall problem-solving performance. Reasoning Paths Optimization does not rely on large-scale human-annotated rationales or outputs from closed-source models, making it scalable and data-efficient. We focus on multi-step reasoning tasks, such as math word problems and science-based exam questions. The experiments demonstrate that our framework significantly enhances the reasoning performance of large language models, with up to 3.1% and 4.3% improvement on GSM8K and MMLU (STEM) respectively. Our data and code can be found at https://reasoning-paths.github.io.

1 Introduction

Large language models (LLMs) have shown remarkable proficiency in following instructions and reasoning (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023b; Jiang et al., 2023). Analogous to human cognitive processes, chain-of-thought prompting guides models to reason step-by-step before producing the final answer (Wei et al., 2022), significantly boosting their reasoning

capabilities and demonstrating exceptional performance across a wide array of tasks (Wang et al., 2023b; Chung et al., 2024). Despite these advancements, LLMs still exhibit limitations in scenarios that require more complex reasoning (Zhong et al., 2024).

As shown in Figure 1, the step-by-step reasoning path of the model is at risk of diverging to unfavorable branches that contain mistakes, thus reducing the chance of reaching the correct solution. While such mistakes may not immediately lead to the wrong answer, they can compound and derail the reasoning process (Ling et al., 2023). Furthermore, this challenge is amplified for more complex problems such as competition-level math questions (Hendrycks et al., 2021b) that require long reasoning paths to solve. Hence, there is a need to address this challenge by encouraging the models to generate the correct reasoning path while avoiding the unfavorable branches.

To ensure a trustworthy answer derivation process, prior studies have explored a range of methods, encompassing both prompting and fine-tuning techniques. Prompting methods repeatedly sample from LLMs for the same question and employ a voting mechanism to select the most accurate reasoning step among several alternatives. Such voting mechanisms can be applied at the final stage of the process, as demonstrated in Self-Consistency (Wang et al., 2023c), or at every intermediate step, as illustrated in Tree-of-Thought (Yao et al., 2023a). Yao et al. (2023b) shows that leveraging external environmental feedback could remind LLMs of some potential errors within their reasoning process, which potentially prevents these errors from affecting subsequent steps. However, the prompting methods generally demand extensive token usage to explore multiple reasoning paths from LLMs and integrate feedback from the environment. This causes a significant computational cost and huge execution latency.
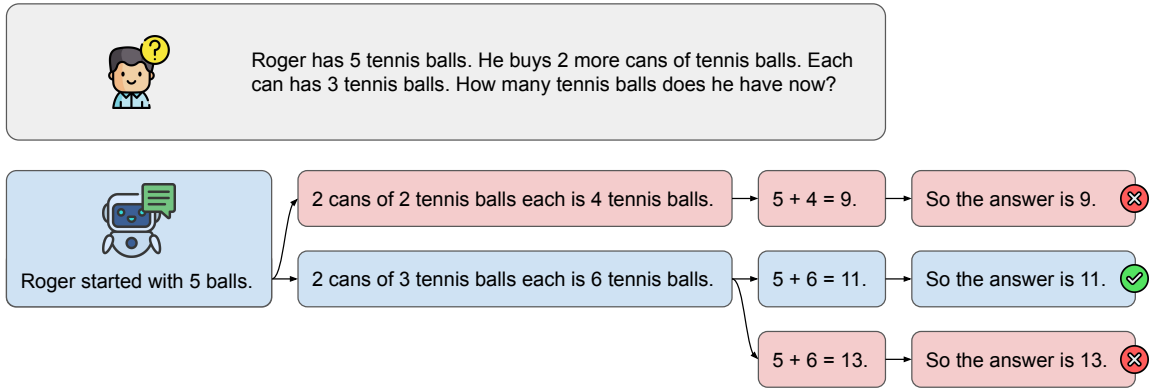

16763

Figure 1: An example of how the reasoning path of the model can easily diverge to unfavorable branches that fail to reach the correct solution. While we show a simplified example here, the challenge is amplified for more complex questions that require longer reasoning paths.

Alternatively, fine-tuning methods can directly enhance the reasoning capability of LLMs without exhaustive prompting engineering. Among these methods, reinforcement learning from human feedback (RLHF) (Christiano et al., 2017a; Stiennon et al., 2020; Ouyang et al., 2022), which involves training a reward model to optimize LLMs, has shown considerable effectiveness in aligning LLMs. This approach further spurs the development of subsequent works focused on preference optimization, such as DPO (Rafailov et al., 2023) and SimPO (Meng et al., 2024), which has gained widespread practical adoption due to its simplicity and stability. However, it has been observed that these preference optimization algorithms may be less effective or even detrimental to tasks requiring in-depth reasoning (Meng et al., 2024). We hypothesize that these optimization methods may indiscriminately target the entire reasoning path as problematic, whereas, as indicated in Figure 2, errors in reasoning often occur at specific steps and affect only the subsequent erroneous branches.

To address the challenge of LLMs committing mistakes that can derail their reasoning paths, we introduce Reasoning Paths Optimization, a novel framework designed to explore and learn from varied reasoning paths. As illustrated in Figure 2, our approach initiates by generating a reference reasoning path for each question that can reach the correct answer via chain-of-thought prompting. Following this, we explore various solution branches emanating from each step in the reference path. With the reference reasoning paths and the potential solution branches explored, we optimize the model from two critical angles: (1) The model should generate the reference reasoning path with a high probability.

(2) The model should favor all potential branches leading to the correct answer over those that do not. To achieve the optimization, we propose a reference loss that maximizes the likelihood of generating the reference reasoning path and an exploration loss that provides contrastive feedback over each pair of favorable and unfavorable branches. As a result, we can explore the diverse mistakes the model is liable to produce, and reduce their occurrence by aligning the models to the correct reasoning path.

Experimental results on math-based reasoning tasks such as GSM8K (Cobbe et al., 2021a) and MATH (Hendrycks et al., 2021b) demonstrate the effectiveness of our approach compared to strong baselines. In addition, we show that Reasoning Paths Optimization can generalize beyond math tasks to improve reasoning performance on the science, technology, engineering, and math (STEM) subset of the MMLU (Hendrycks et al., 2021a) exam question dataset. Notably, the experiments show up to 3.1% and 4.3% improvement compared to the high-performing baseline on GSM8K and MMLU (STEM) datasets respectively.

2 Reasoning Paths Optimization

2.1 Task Formulation

In this work, we focus on problems that require multiple steps to arrive at the final answer or produce the final result, such as math word problems (Cobbe et al., 2021b; Hendrycks et al., 2021b). Thus, we provide a concrete task formulation in this section. Given a question Q posed in natural text, the goal is to produce the final answer A in natural text. The model is assumed to go through several reasoning steps $S_1, S_2, \ldots, S_n$ to arrive at
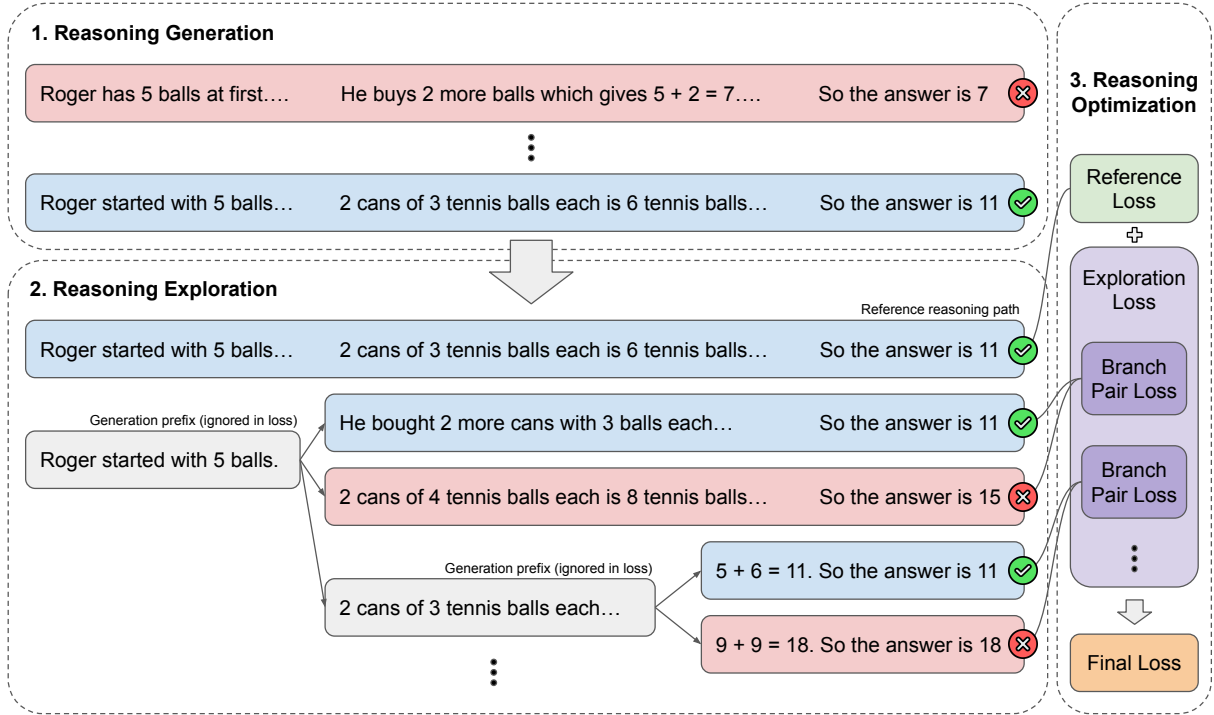
Figure 2: An overview of our Reasoning Paths Optimization framework for exploring and learning over diverse reasoning paths.

the final answer A. The reasoning path P is defined as the sequence of these steps:

$$P = (S_1, S_2, \ldots, S_n) \qquad (1)$$

where each S_i is a natural language sentence and the last step S_n should contain the answer A. Although the model could generate the correct final answer either devoid of a reasoning path or containing faulty reasoning steps, the findings and analysis presented in Section 3 suggest that such an outcome is improbable. To determine the separation points between different steps in the generation, we utilize the punctuation-based sentence splitting tool from NLTK, as the reasoning process follows a natural language structure.

2.2 Framework Overview

Large language models are capable of reasoning step-by-step to enhance their problem-solving abilities. However, they often fall short when faced with more challenging problems, committing mistakes that derail their reasoning paths. We believe this issue arises from the large solution space, where multiple reasoning paths can lead to the correct final answer, but each step carries the risk of branching into errors. To address this, we propose a specialized training framework that jointly considers

diverse reasoning paths for a given problem. Our approach encourages favorable branches at each reasoning step while penalizing unfavorable ones. This framework, which we call Reasoning Paths Optimization (RPO), consists of three main stages as shown in Figure 2:

1. Generation: The generation stage aims to elicit correct reasoning steps from the base model to serve as reference reasoning paths. This eliminates the need for acquiring ground-truth reasoning path annotations.

2. Exploration: To effectively explore the potential solution space to a given problem, this stage progressively creates branches from each step along reference reasoning paths. As a result, we can obtain multiple favorable and unfavorable reasoning branches, which will be used to provide contrastive feedback to the model.

3. Optimization: This final stage aggregates and optimizes according to the reference reasoning paths and explored branches to enhance the innate reasoning ability of the base model. Thus, our framework aims to improve the overall reasoning ability of large language models.

2.3 Reasoning Generation

While training with explanations or step-by-step reasoning paths (Mukherjee et al., 2023) can im-

prove the reasoning performance of language models, it is labor-intensive and costly to annotate such data. Hence, our framework begins with a reasoning generation stage that automatically generates the reference reasoning paths. Concretely, given a problem question Q, we use chain-of-thought prompting (Wei et al., 2022) to generate reasoning paths. The chain-of-thought demonstration input D_{CoT} consists of m ground-truth examples, where each example is a pair consisting of a problem question and its corresponding reasoning path.

Let M be the base model, and we sample the reference reasoning path P by prompting the model with the chain-of-thought demonstration D_{CoT} and the given question Q. We use temperature sampling (Fan et al., 2018) with a fixed temperature T:

$$P \sim M(D_{CoT}, Q | T) \qquad (2)$$

We consider the generated path as correct if it concludes with a correct answer. Therefore, we define the following function $\mathcal{F}$ to verify if the last step $S_n \in P$ contains the ground-truth answer A:

$$\mathcal{F}(P) = \begin{cases} 1 & \text{if } A \in S_n \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

If the outputs are incorrect, i.e., $\mathcal{F}(P_i) = 0$, we repeat the sampling and verification process until $\mathcal{F}(P_i) = 1$ with a cap of 10 attempts, i.e., $i \le 10$. If no suitable path is obtained after multiple attempts, we deem that this problem is far beyond the ability of the model and remove it from the training set. Thus, we obtain an initial dataset D_{init} which contains the original questions, the ground-truth answers, and model-generated reference paths.

2.4 Reasoning Exploration

To consider potential mistakes that can occur from each reasoning step, the exploration stage of our framework explores multiple branches at each step. Concretely, given the problem Q, chain-of-thought demonstration, and previous steps of the generated reasoning path $P_{1:i-1} = (S_1, S_2, \ldots, S_{i-1})$, we use temperature sampling (Fan et al., 2018) to obtain diverse branches from the current point in the reasoning path:

$$B_i \sim M(D_{CoT}, Q, P_{1:i-1} | T), \qquad (4)$$

where each branch $B_i = (S'_i, S'_{i+1}, \ldots, S'_l)$ should contain the current step up to the final step. We aim to obtain a favorable branch B_i^+ and an unfavorable branch B_i^- where the favorable branch leads to the correct final answer, and the unfavorable branch does not:

$$\mathcal{F}(B_i^+) = 1, \quad \mathcal{F}(B_i^-) = 0 \qquad (5)$$

To achieve this, we iteratively sample multiple branches starting at each step S'_i and verify each one using the function $\mathcal{F}$, until we obtain one favorable branch and one unfavorable branch, thus forming a reasoning branch pair (B_i^+, B_i^-). However, if we are unable to form a branch pair after sampling at most ten branches, the problem is removed from the training set. This ensures that the training data only includes problems where the model can potentially learn from contrasting between the favorable and unfavorable branches of the reasoning path.

2.5 Reasoning Optimization

To optimize the base model M, we consider both the reference reasoning path P generated in Sec. 2.3 and the reasoning branch pairs (B_i^+, B_i^-) explored in Sec 2.4. Concretely, we encourage the model to produce a higher likelihood over the reference reasoning path. This is achieved by applying standard causal language modeling loss (Bengio et al., 2000) on the reference reasoning path P, conditioned on the input question Q:

$$\mathcal{L}_{ref} = -\log Pr_M(P \mid Q) \qquad (6)$$

Regarding the branch pair, the comparison between them may reveal the proper direction that guides the model's optimization. Therefore, we define a branch pair loss that provides contrastive feedback between the favorable and unfavorable branches. To formulate the branch pair loss in our framework, we can leverage preference-based objectives from existing work, such as the direct preference (Rafailov et al., 2023) or the odds-ratio objective (Hong et al., 2024). In this work, we mainly focus on the objective proposed by Hong et al. (2024) due to its simplicity and empirical effectiveness. Concretely, the branch pair loss $\mathcal{L}_{bp,i}$ at the i-th step can be computed as the log odd-ratio between the favorable branch B_i^+ and unfavorable branch B_i^-, conditioned on the input question Q and reference path P:

$$\mathcal{L}_{bp,i} = \log \frac{\text{odds}_M(B_i^+ \mid Q, P)}{\text{odds}_M(B_i^- \mid Q, P)} \qquad (7)$$

The odds of generating a branch can be computed as the ratio between the probability of generating the branch and the probability of not generating it, conditioned on the input question Q and the previous steps $P_{1:i-1}$ of the reference path:

$$\mathbf{odds}_M(B_i \mid Q, P) = \frac{Pr_M(B_i \mid Q, P_{1:i-1})}{1 - Pr_M(B_i \mid Q, P_{1:i-1})} \tag{8}$$

Thus, we can aggregate the loss over the previously explored branch pairs corresponding to each step in the reasoning path:

$$\mathcal{L}_{exp} = \frac{1}{n-1} \sum_{i=1}^{n} -\log \sigma(\mathcal{L}_{bp,i}) \tag{9}$$

where there are n steps in the reasoning path. We follow Hong et al. (2024) to apply the log-sigmoid function $\log \sigma$ on the log odds-ratio for optimization purposes. Finally, the overall loss $\mathcal{L}_{RPO}$ in our framework is represented as the combination of the reference path loss $\mathcal{L}_{ref}$ and the exploration loss $\mathcal{L}_{exp}$ which provides contrastive feedback over the explored branch pairs:

$$\mathcal{L}_{RPO} = \mathcal{L}_{ref} + \lambda \cdot \mathcal{L}_{exp} \tag{10}$$

where λ is a hyperparameter weight, which intuitively balances between optimizing on the reference reasoning path, and the explored branches.

We would like to clarify that we compute the loss only on the output tokens. In this case, the output tokens only consist of the incorrect last part, while the correct prefixes serve as the input tokens, which are excluded from the loss calculation as shown in Figure 2. Specifically, the reasoning exploration stage in our framework first collect branch pairs from each step along a reference path, then aggregates the branch pair losses conditioned on the input question and the previous steps of the reference path. Therefore, the common prefix between the favorable and unfavorable branch is excluded in the loss calculation.

3 Experiments

3.1 Datasets

As we focus on enhancing the step-by-step reasoning ability of large language models, we evaluate our approach on datasets of various difficulty levels, including GSM8K (Cobbe et al., 2021b) for math word problems and MATH (Hendrycks et al., 2021b) for competition-level mathematics. We use the original training, validation, and testing data splits for our training and evaluation setup. On the other hand, we also include the MMLU (Hendrycks et al., 2021a) exam question dataset to evaluate the effectiveness of our approach in other domains. However, as many of the exam questions focus on world-knowledge and do not require multi-step reasoning, we extract a subset covering 3375 questions in the science, technology, engineering, and math (STEM) domains, and denote this as the MMLU-STEM dataset. The dataset details can be found in Appendix A.2.

Note that our Reasoning Paths Optimization framework does not necessitate large-scale annotated reasoning paths for training LLMs. On the contrary, for each task, we only need a small number of reasoning demonstrations for implementing CoT prompting, which is easy to obtain. Specifically, we randomly select four questions from the training data and use their ground-truth reasoning path as CoT demonstrations during the reasoning generation stage. For the remaining procedure, Reasoning Paths Optimization only involves the ground-truth answer to verify the correctness of the explored branch. We include the prompt examples in Appendix A.3.

3.2 Implementations

To evaluate our approach, we implement Mistral-7B and LLaMA-3-8B as our base models, which are recent and popular foundation large language models in the Mistral (Jiang et al., 2023) and LLaMA (Touvron et al., 2023a) model families respectively. To our knowledge, these are the leading foundation models in this parameter size category at the time of writing. To investigate how our approach affects models of different training stages, we also include experiments show that our framework also benefits the LLaMA-3-8B-Instruct version in Appendix A.4, which has undergone general instruction-tuning (Touvron et al., 2023a) to enhance performance in many aspects. Due to computational resource constraints, we are unfortunately unable to train larger model versions such as LLaMA-3-70B in this work. To avoid potential confounding factors, we do not evaluate on models that already have extensive math-specific training, such as Llemma (Azerbayev et al., 2024). To train
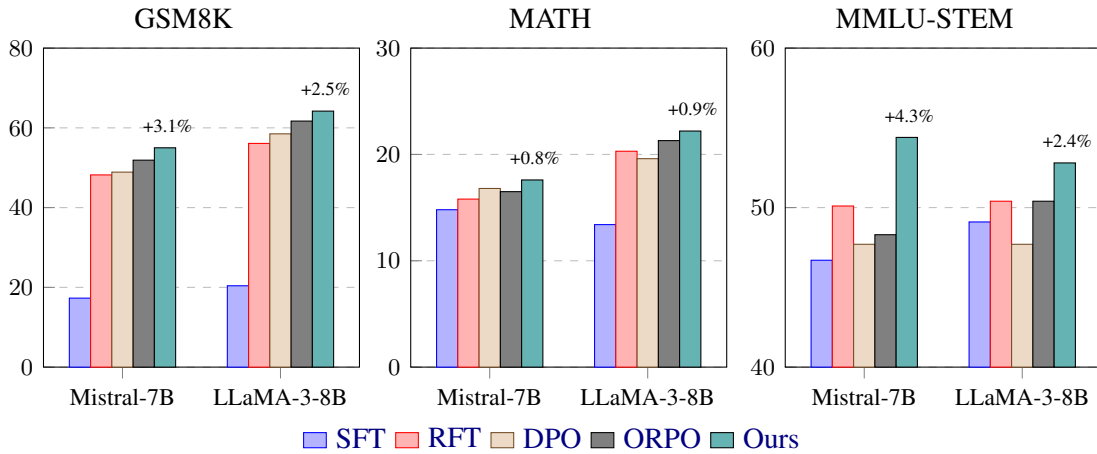
Figure 3: Main results showing the evaluation accuracy (%) of different training methods on math reasoning questions in GSM8K and MATH, and science-based exam questions in MMLU-STEM. We also indicate the improvement of our method compared to the highest-performing baseline.

the models, we use LoRA fine-tuning (Hu et al., 2022) with a fixed batch size of 8 and a learning rate of 5e-5. More training details and hyperparameters can be found in the Appendix A.1. To sample multiple outputs from the models, we use a fixed sampling temperature of 0.5. For evaluation, we use greedy decoding for generation, and the accuracy metric for scoring.

3.3 Comparison Methods

To demonstrate the effectiveness of our approach, we compare against strong baselines including reasoning-specific training methods and preference-based optimization methods:

1. Supervised Fine-Tuning (SFT): As a supervised baseline, we consider the case of not using any reasoning paths for training, and only training the model to directly generate the ground-truth final answer.

2. Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2024): We include RFT as a strong baseline for supervised training, which leverages the model to self-generate reasoning paths for training. We note that this approach is analogous to the reasoning generation stage in our framework, which aims to overcome the data limitation of not having ground-truth reasoning paths.

3. Direct Preference Optimization (DPO) (Rafailov et al., 2023): As our method contrasts the favorable and unfavorable reasoning branches, it is similar in motivation

to DPO which provides the model with contrastive feedback.

4. Odds-Ratio Preference Optimization (ORPO) (Hong et al., 2024): Lastly, we compare against ORPO which proposed the odds ratio objective for preference-based optimization. The main difference between our approach and ORPO is that Reasoning Paths Optimization is a holistic framework specifically designed for reasoning-based tasks; We consider that reasoning mistakes are liable to occur at any step in the reasoning path, and hence explore the possible solution paths which are necessary to provide contrastive feedback over diverse reasoning branch pairs.

To ensure a fair comparison between different methods, we implement the data setting such that each method uses all viable training samples. For instance, SFT uses all the training samples as the data setting stipulates that all samples contain the question and ground-truth final answers. On the other hand, RFT uses only the samples for which the model can generate at least one correct reasoning path, and the preference-based methods DPO and ORPO use only the samples for which the model can generate at least one correct reasoning path and one incorrect reasoning path. Similar to our approach, the baselines other than SFT use a fixed temperature for sampling reasoning paths with chain-of-thought prompting. If the model is unable to generate a correct reasoning path after sampling a maximum of ten times, the given question is removed from the training set.
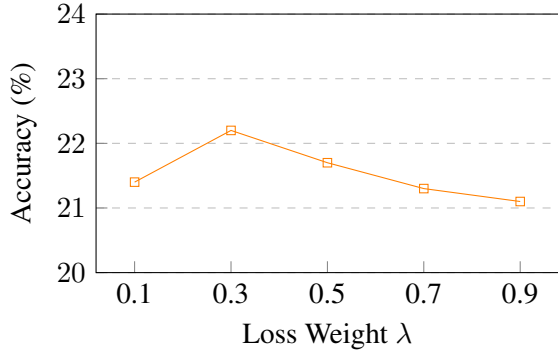
16768

Figure 4: The effect of exploration loss weight on the MATH dataset performance for LLaMA-3-8B.

Figure 5: Performance with respect to reasoning path length on the MATH dataset for LLaMA-3-8B.

3.4 Main Results

To demonstrate the effectiveness of Reasoning Paths Optimization, we compare with strong baselines as shown in Figure 3. We observe that our approach shows consistent improvements in performance on different datasets and models. Particularly when trained on top of Mistral-7B, our approach can achieve up to 3.1% and 4.3% improvement compared to the highest-performing baseline on GSM8K and MMLU-STEM respectively. Given that MATH is a relatively difficult task, the base models may struggle to generate the correct paths, thereby limiting the effectiveness of path-based methods. Nevertheless, our approach can still improve other baselines, which shows that our approach can more effectively learn from the explored reasoning paths. On the other hand, we find that SFT performance is lower compared to the other methods trained on self-explored reasoning paths. This indicates that while it is possible for the model to directly generate the answer without any reasoning steps, it is less effective for more complex reasoning problems.

We further investigate the performance of our method on commonsense and general reasoning tasks in Appendix A.5. These tasks typically consist of straightforward questions that do not require lengthy reasoning steps, which may possibly contribute to the high SFT performance. Nevertheless, when the model is prompted to engage in step-by-step reasoning, our framework outperforms other preference optimization approaches, demonstrating its effectiveness in multi-step thinking.

3.5 Effect of Exploration Weight

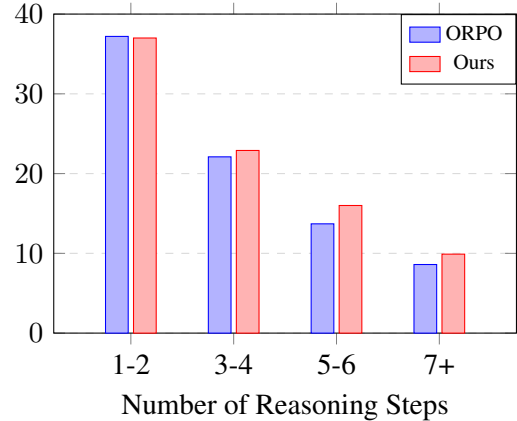To investigate the effect of reasoning exploration within our framework, we conduct an analysis on

the loss weight λ. Specifically, a lower value of λ would place greater emphasis on the supervised loss over the reference path which leads to the correct answer. On the other hand, a higher value would place greater weight on the explored branches during training, which contrasts between the favorable and unfavorable branches arising from each reasoning step. As shown in Figure 4, we find that having a very low λ value leads to sub-optimal results, as it does not place enough emphasis on the reasoning exploration. On the other hand, we also find that placing too much weight on exploration also does not benefit the training, as it is still necessary to ground the model sufficiently in the reference reasoning path. Thus, this shows that we need to balance between the optimization of the reference reasoning path and the other possible exploration branches.

3.6 Analysis On Reasoning Path Length

To investigate the effectiveness of our approach with respect to the reasoning complexity, we analyze the evaluation accuracy in Figure 5. Compared to ORPO (Hong et al., 2024) which is the highest-performing baseline, we observe benefits from Reasoning Paths Optimization for longer reasoning paths. This may indicate that our approach can effectively reduce the occurrence of mistakes when solving questions that require more complex reasoning. In future work, we believe that potential directions include more diverse applications and tasks, such as code generation (Chen et al., 2021; Austin et al., 2021), and code-augmented reasoning (Gao et al., 2023; Li et al., 2023a).

16769

Question	It takes John 5 minutes to get to the first side of the Rubik's cube. The second and third sides each take twice that long. The last 2 sides each take half as long as the first, and once you get the 5th side you have all 6 sides. How long does the Rubik's cube take?
SFT	25
RFT	5 + 2*5 + 2*5 + 2*5 + 2*5 + 2*5 = 5 + 10 + 10 + 10 + 10 + 10 = 55 minutes. So the answer is $\boxed{55}$ minutes.
DPO	5 + 10 + 10 + 5 + 5 = 35 minutes. So the answer is $\boxed{35}$ minutes.
ORPO	5 + 2*5 + 2*5 + 2*5 + 2*5 + 2*5 = 5 + 10 + 10 + 10 + 10 = 55 minutes. So the answer is $\boxed{55}$ minutes.
RPO	The first side takes 5 minutes. The second and third sides each take 10 minutes. The last 2 sides each take 2.5 minutes. The total time is 5 + 10 + 10 + 2.5 + 2.5 = 30 minutes. So the answer is $\boxed{30}$ minutes.

Table 1: A qualitative comparison between GSM8K outputs after training LLaMA-3-8B with different methods.

Training Method	GSM8K (Text)	GSM8K (Code)
ORPO	61.7	61.6
RPO (Ours)	**64.2**	**63.4**

Table 2: Analysis of performance benefits for text-based reasoning as our main setting, and code-based reasoning through python programs. Experiments are conducted using LLaMA-3-8B.

Method	GSM8K	MATH	MMLU
DPO	58.5	19.6	47.7
ORPO	61.7	21.3	50.4
Ours (w/ direct-preference)	60.7	21.4	52.8
Ours (w/ odds-ratio)	64.2	22.2	52.8

Table 3: Performance comparison on GSM8K, MATH, and MMLU-STEM datasets for different contrastive objectives in our framework using LLaMA-3-8B.

Exploration	GSM8K
ORPO	61.7
Ours (w/ first correct as reference path)	64.2
Ours (w/ random one correct as reference path)	63.5
Ours (w/ random three correct as reference paths)	65.2

Table 4: Performance comparison on GSM8K for different reference paths using LLaMA-3-8B.

3.7 Case Study

To examine the qualitative benefits, Table 1 shows an example of outputs produced by different methods for the same question. While SFT provides an incorrect and over-simplified answer without showing any working, RFT, DPO, and ORPO directly calculate without explanation and thus make mistakes at the very first step. In contrast, the detailed breakdown of steps shows that RPO not only arrives at the correct answer but does so through a coherent process.

3.8 Code-Based Reasoning

Beyond reasoning in natural language works such as PAL (Gao et al., 2023) have shown that large language models can be prompted to solve reasoning problems with code. To this end, we have conducted an analysis to show that our framework can also generalize to code-based reasoning. Concretely, in our reasoning generation stage, instead of generating text-based reasoning paths, we prompt the model with code demonstrations to generate a python program, which is executed to obtain the output answer. As shown in Table 2, we find similar benefits for text-based reasoning and code-based reasoning compared to ORPO, which is our strongest baseline.

3.9 Effect of Contrastive Objectives

To demonstrate the robustness of our framework, we have conducted additional experiments using different objectives to contrast between favorable and unfavorable paths. Specifically, we show that the odds-ratio objective (Hong et al., 2024) in our branch pair loss can be easily replaced with the direct preference objective (Rafailov et al., 2023) for the branch pair loss in Equation 7. As shown in Table 3, the consistent benefit across different objectives demonstrates that our framework is robust and outperforms the respective baselines.

3.10 Effect of Reference Paths

In our exploration stage, we use the first reasoning path with the correct answer as the reference path. However, the correct answer can often be achieved via different paths. To analyse the effect of different reference paths, we select a random path with the correct answer after sampling 10 times. Results in Table 4 show that our method remains effective even with this variation, demonstrating its robustness across different reference paths. In addition, we analyse the effect of using more reference paths, eg, three correct reference paths. The results show that our approach can scale to multiple reference paths to further enhance performance.

4 Related Work

Alignment and Preference-Based Optimization
Reinforcement learning from human feedback (RLHF) (Christiano et al., 2017b; Ouyang et al., 2022) is a popular technique that aligns large language models with human values and preferences. During RLHF, a separate reward model is trained to provide scalar value feedback, which is passed to fine-tune LLMs with PPO algorithm (Schulman et al., 2017; Ziegler et al., 2019). However, PPO is known to be complex and unstable (Zheng et al., 2023), and the multi-stage training of a reward model and a policy model is also challenging (Meng et al., 2024). Recently, several techniques, including DPO (Rafailov et al., 2023), IPO (Azar et al., 2023), SimPO (Meng et al., 2024), and ORPO (Hong et al., 2024), have been proposed to eliminate the need for a reward model, which significantly stabilize and simplify the training process. They make pairwise comparisons between two responses generated by the models and push the model to assign a higher likelihood to the favorable response over the unfavorable one. However, these preference optimization methods indiscriminately compare the two responses in their entirety, overlooking the fact that errors in multi-step reasoning tasks arise only at specific steps and their subsequent branches. In this work, we propose Reasoning Paths Optimizationto make pairwise comparisons emanating from each intermediate step.

Multi-step Reasoning in Language Models
Large language models are capable of solving reasoning tasks by generating solutions in a step-by-step manner (Nye et al., 2022; Wei et al., 2022; Kojima et al., 2022; Fu et al., 2023; Chu et al., 2024). For example, Wei et al. (2022) and Kojima et al. (2022) demonstrate that by guiding the model to generate the reasoning steps before generating the final answer, the multi-step reasoning capabilities of LLMs could be effectively elicited. However, LLMs are prone to producing errors during the reasoning process, especially for complex multi-step reasoning tasks. To mitigate mistakes in the reasoning steps, a straightforward way is to verify the reasoning paths step-by-step. This encourages further investigations on process supervision. Uesato et al. (2022) and Lightman et al. (2024) collect human feedback labels for step-level solutions to verify the intermediate steps generated by reasoning models. Recent studies (Li et al., 2023b; Wang et al., 2024a,b) propose to annotate the step-wise labels automatically to prevent costly human annotations. These methods focus on training the verifiers (i.e., reward models). In contrast, we apply process supervision to preference optimization methods, without requiring a separate reward model.

Path Exploration in Artificial Intelligence
The exploration of diverse paths has been widely used to improve the performance of complex tasks in the field of artificial intelligence. AlphaGo (Silver et al., 2016) uses Monte Carlo Tree Search (Kocsis and Szepesvári, 2006) to explore a large space of possible moves. Similarly, in the field of LLMs, Yao et al. (2023a) leverage Tree-of-Thought prompting to explore possible solution space from LLMs. Other works (Feng et al., 2023; Xie et al., 2023) also design tree-based decoding strategies to search for the optimal solution. In the area of reasoning tasks, previous works have explored using self-sampled solutions for training (Ni et al., 2023) and tree search for path generation (Golovneva et al., 2023). Inspired by these works, we explore the diverse solution space generated by language models. Furthermore, we optimize the models with contrastive feedback from both favorable and unfavorable branches during training. Inspired by these works, we explore the diverse solution space generated by the models. Furthermore, we optimize LLMs with both favorable and unfavorable branches during training.

5 Conclusion

In this paper, we introduced a novel training framework called Reasoning Paths Optimization (RPO) to enhance the step-by-step reasoning capabilities of LLMs. Our approach addresses the challenge of complex problem-solving tasks, where each reasoning step carries the risk of diverging into errors. RPO considers diverse reasoning branch pairs and encourages favorable branches at each reasoning step while penalizing unfavorable ones. Our framework is scalable, as it does not rely on large-scale human-annotated rationales. Instead, it leverages the model's own generated reasoning paths, making it adaptable to multi-step reasoning tasks such as math word problems. Through extensive experiments on datasets of varying difficulties, our framework provides an effective approach to enhance reasoning, paving the way for more reliable and accurate problem-solving in complex scenarios.

Acknowledgment

Limitations

Our framework relies on the model's ability to generate correct reasoning paths during the training phase. If the base model is significantly underperforming, it may struggle to generate the necessary correct paths, thereby limiting the effectiveness of our approach. To provide performance insights beyond accuracy, we also report the Inter. F1 metric in Appendix A.6, which demonstrates that the reasoning paths generated after training with our method is more consistent with the ground-truth reasoning paths in GSM8K. Although the process of generating and exploring multiple reasoning paths for each problem is more computationally intensive, we note that this is a one-time cost during training. Hence, we believe that this is a worthwhile trade-off to enhance performance, which can be amortized over many inference cases.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *Preprint*, arXiv:2310.12036.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. *Preprint*, arXiv:2310.10631.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017a. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017b. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2024. Navigate through enigmatic labyrinth a survey of chain of thought reasoning: Advances, frontiers and future. *Preprint*, arXiv:2309.15402.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias

Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021b. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Olga Golovneva, Sean O'Brien, Ramakanth Pasunuru, Tianlu Wang, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Pathfinder: Guided search over multi-step reasoning paths. *Preprint*, arXiv:2312.05180.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *Preprint*, arXiv:2403.07691.

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

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. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, page 282–293, Berlin, Heidelberg. Springer-Verlag.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Chengshu Li, Jacky Liang, Fei Xia, Andy Zeng, Sergey Levine, Dorsa Sadigh, Karol Hausman, Xinyun Chen, Li Fei-Fei, and brian ichter. 2023a. Chain of code: Reasoning with a language model-augmented code interpreter. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Preprint*, arXiv:2405.14734.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. 2023. Learning math reasoning from self-sampled correct and partially-correct solutions. In *The Eleventh International Conference on Learning Representations*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2022. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.

2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande. *Communications of the ACM*, 64:99 – 106.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv preprint*, abs/1707.06347.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten,

Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023a. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

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

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, L. Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *ArXiv*, abs/2211.14275.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Jianing Wang, Qiushi Sun, Nuo Chen, Xiang Li, and Ming Gao. 2023b. Boosting language models reasoning with chain-of-knowledge prompting. *arXiv preprint arXiv:2306.06427*.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

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

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. 2024b. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *Preprint*, arXiv:2402.02658.

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

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. Scaling relationship on learning mathematical reasoning with large language models.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. 2023. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, Chao Cao, Hanqi Jiang, Hanxu Chen, Yiwei Li, Junhao Chen, Huawen Hu, Yihen Liu, Huaqin Zhao, Shaochen Xu, Haixing Dai, Lin Zhao, Ruidong Zhang, Wei Zhao, Zhenyuan Yang, Jingyuan Chen, Peilong Wang, Wei Ruan, Hui Wang, Huan Zhao, Jing Zhang, Yiming Ren, Shihuan Qin, Tong Chen, Jiaxi Li, Arif Hassan Zidan, Afrar Jahin, Minheng Chen, Sichen Xia, Jason Holmes, Yan Zhuang, Jiaqi Wang, Bochen Xu, Weiran Xia, Jichao Yu, Kaibo Tang, Yaxuan Yang, Bolun Sun, Tao Yang, Guoyu Lu, Xianqiao Wang, Lilong Chai, He Li, Jin Lu, Lichao Sun, Xin Zhang, Bao Ge, Xintao Hu, Lian Zhang, Hua Zhou, Lu Zhang, Shu Zhang, Ninghao Liu, Bei Jiang, Linglong Kong, Zhen Xiang, Yudan Ren, Jun Liu, Xi Jiang, Yu Bao, Wei Zhang, Xiang Li, Gang Li, Wei Liu, Dinggang Shen, Andrea Sikora, Xiaoming Zhai, Dajiang Zhu, and Tianming Liu. 2024. Evaluation of openai o1: Opportunities and challenges of agi. *Preprint*, arXiv:2409.18486.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *ArXiv preprint*, abs/1909.08593.

A Appendix

A.1 Training and Hyperparameter Details

We list the hyperparameter and training details in Table 5. Note that we use the validation accuracy of MATH with LLaMA-3-8B to select the loss weight $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and use it for all datasets. To ensure the diversity in our reasoning exploration stage, we remove duplicate reasoning paths and branches before training. To maintain fairness between different training methods, we ensure that each method uses at most one accepted or reference reasoning path that reaches the correct answer for each question. For preference-based methods, we ensure that each method uses at most one accepted reasoning path, and one rejected reasoning path that reaches the wrong answer for each question. Similarly, our approach uses at most one reference reasoning path for each question. To be fair to DPO which typically follows an SFT training stage, we include the SFT loss over the accepted reasoning path during training, which is a joint loss with the DPO objective. Hence, all methods in this paper utilize a single training stage.

As shown in Table 6, all training methods use a similar amount of training data in terms of reasoning paths. Note that the number of samples used for each model is different as the samples are filtered based on the correctness of model outputs.

A.2 Dataset Details

For GSM8K and MATH, we use the original training and testing splits. For MMLU (STEM), we specifically use the STEM subset for the following subcategories of questions:

- abstract_algebra
- astronomy
- college_biology
- college_chemistry
- college_computer_science
- college_mathematics
- college_physics
- computer_security
- conceptual_physics
- electrical_engineering
- elementary_mathematics
- high_school_biology
- high_school_chemistry
- high_school_computer_science
- high_school_mathematics
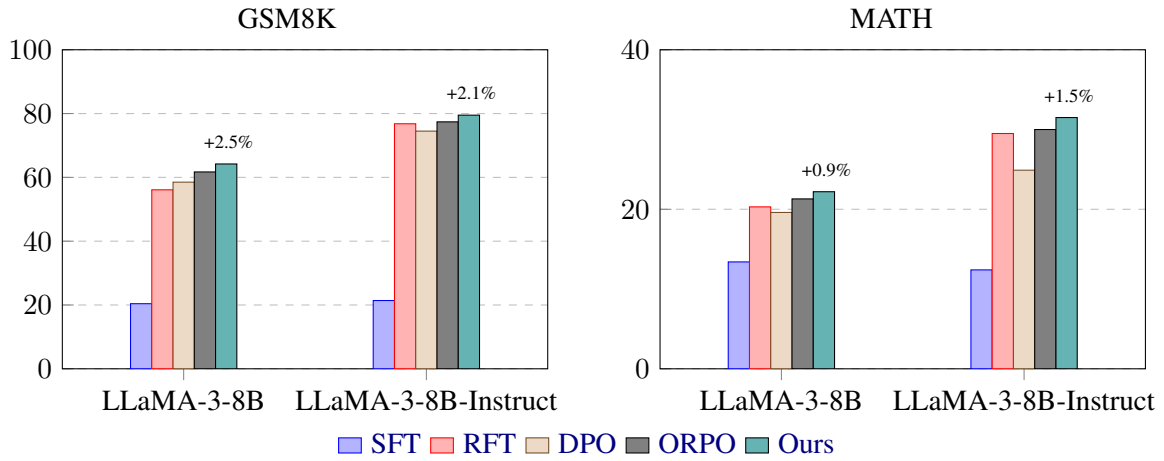- high_school_physics

16775

Figure 6: Main results showing the evaluation accuracy (%) of different training methods on math reasoning datasets. We also indicate the improvement of our method compared to the highest-performing baseline.
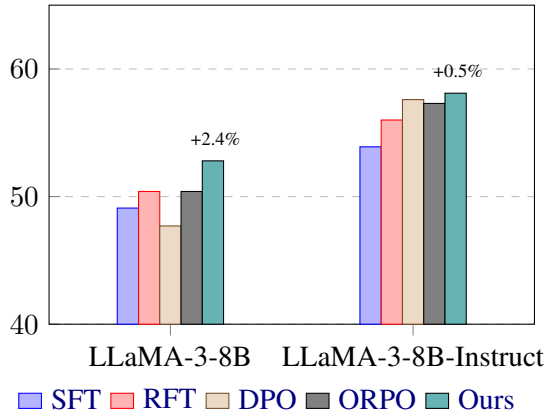

Figure 7: Additional results showing the evaluation accuracy on science, technology, engineering, and math questions in MMLU (Hendrycks et al., 2021a). We also indicate the improvement of our method compared to the highest-performing baseline.

Method	LLaMA-3-8B	Mistral
SFT	7473 answers	7473 answers
RFT	6417 reasoning paths	5922 reasoning paths
DPO	5667 reasoning path pairs	5535 reasoning path pairs
ORPO	5667 reasoning path pairs	5535 reasoning path pairs
RPO	5752 reference paths	5600 reference paths

Table 6: Training data comparison for different methods on GSM8K dataset using different models.

A.3 Prompting

For our reasoning generation stage as well as the baselines of RFT, DPO, and ORPO, we use chain-of-thought prompting to generate training reasoning paths. Note that we use 4-shot prompting for all settings and methods as shown below:

GSM8K CoT Prompt

```
Question: There are 180 days in a school
year. A senior can skip their final exams
if they miss 5% or less of the school year.
Hazel has missed 6 days of school due to
illness. How many more days can she miss
and still not have to take her exams?

Answer: There are 180 days in the school
year and she can miss up to 5% so that's
180*.05 = 9 days\nHazel has been sick 6
days already and she can only miss 9 days
or less so she can miss 9-6 = 3 more days.
So the answer is \\boxed{3} days.

Question: Several birds were sitting in
the branches of a crape myrtle tree.
There were three times more cardinals
than bluebirds, but half as many swallows
```

Loss weight λ	0.3
Lora rank	8
Learning rate	5e-5
Batch size	8
Training epochs	3
Hardware	Single A800 (80GB)

Table 5: Hyperparameter and training details.

- high_school_statistics

- machine_learning

We thus create a train-test split of the STEM questions, contain 3000 training and 375 testing samples.

as bluebirds. If there were 2 swallows, what is the total number of birds in the crape myrtle tree?

Answer: With half as many swallows as bluebirds, there are 2*2=4 bluebirds. With three times more cardinals than bluebirds, there are 3*4=12 cardinals, If there were 2 swallows, then the total number of birds in the crape myrtle tree is 2+4+12=18 birds. So the answer is \\boxed{18}.

Question: Barry goes to a shop to buy a shirt he'd been admiring for quite some time. He tells the attendant that it's his birthday so she decides to give him a 15% special discount. The price tag on the shirt says $80. How much is he supposed to pay now, considering the special discount?

Answer: 15% of $80 = (15/100)*$80 = $12 The dollar amount of the discount is $12 so he is supposed to pay just $80-$12 = $68. So the answer is \\boxed{$68}.

Question: Nancy wanted to make peanut butter cookies for a family gathering, but her cousin is allergic to peanuts. She decided to make almond butter cookies instead. A jar of almond butter costs three times the amount that a jar of peanut butter does. It takes half a jar to make a batch of cookies. A jar of peanut butter costs $3. How many dollars more does it cost per batch to make almond butter cookies instead of peanut butter cookies?

Answer: A jar of almond butter costs 3 * 3 = $9.\nIt takes half a jar to make a batch of cookies, so it costs 9 / 2 = $4.50 to use almond butter.\nIt costs 3 / 2 = $1.50 to use peanut butter.\nThus, it costs 4.50 - 1.50 = $3 more to make a batch of almond butter cookies than peanut butter cookies. So the answer is \\boxed{$3}.

MATH CoT Prompt

Question: Find the domain of the expression

Method	Precision	Recall	F1
RFT	81.9	75.3	77.5
DPO	80.8	78.9	79.0
ORPO	82.7	78.0	79.5
RPO (Ours)	83.5	79.9	80.9

Table 7: Analysis of the reasoning quality through Inter. Precision, Recall, and F1 metrics on GSM8K.

$\\frac{\sqrt{x-2}}{\sqrt{5-x}}$.}

Answer: The expressions inside each square root must be non-negative. Therefore, $x-2 \ge 0$, so $x\ge2$, and $5 - x \ge 0$, so $x \le 5$. Also, the denominator cannot be equal to zero, so $5-x>0$, which gives $x<5$. Therefore, the domain of the expression is $\\boxed{[2,5)}$.\nSo the final answer is \\boxed{[2,5)}.

Question: If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12,$ then find $\det (\mathbf{A} \mathbf{B}).$

Answer: We have that $\det (\mathbf{A} \mathbf{B}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \\boxed{24}.$ So the final answer is \\boxed{24}.

Question: Terrell usually lifts two 20-pound weights 12 times. If he uses two 15-pound weights instead, how many times must Terrell lift them in order to lift the same total weight?

Answer: If Terrell lifts two 20-pound weights 12 times, he lifts a total of $2\cdot 12\cdot20=480$ pounds of weight. If he lifts two 15-pound weights instead for n times, he will lift a total of $2\cdot15\cdot n=30n$ pounds of weight. Equating this to 480 pounds, we can solve for n:\n\\begin{align*}\n30n&=480\\\\ \Rightarrow\qquad n&=480/30=\\boxed{16} \end{align*} So the final answer is \\boxed{16}.

Question: If the system of equations \n\\begin{align*}\n6x-4y&=a,\\\\ 6y-9x &=b.\n\\end{align*}\nhas a solution

16777

(x, y) where x and y are both nonzero, find $\frac{a}{b},$ assuming b is nonzero.

Answer: If we multiply the first equation by $-\frac{3}{2}$, we obtain $$6y-9x=-\frac{3}{2}a.$$\nSince we also know that $6y-9x=b$, we have $$-\frac{3}{2}a=b\Rightarrow\frac{a}{b}=\boxed{-\frac{2}{3}}.$$
So the final answer is \\boxed{-\\frac{2}{3}}.

MMLU (STEM Subset) CoT Prompt

Question: Which of the following adaptive features would least likely be found in an animal living in a hot arid environment? \n(A) Long loops of Henle to maximize water reabsorption\n(B) Storage of water in fatty tissues\n(C) Large ears to aid in heat dispersion\n(D) Short loops of Henle to maximize water secretion

Answer: To determine which adaptive feature would be least likely found in an animal living in a hot arid environment, let's analyze each option in the context of such an environment: (A) Long loops of Henle to maximize water reabsorption: In a hot arid environment, water is scarce. Animals would benefit from adaptations that help conserve water. Long loops of Henle in the kidneys are an adaptation that allows for maximum reabsorption of water, which is crucial for survival in such conditions. This feature is likely to be found in animals living in arid environments. (B) Storage of water in fatty tissues: Some animals have evolved to store water in their bodies to survive long periods without access to water. Storing water in fatty tissues is an effective way to ensure a supply of water during droughts or dry periods. This is another adaptation that would be beneficial in a hot arid environment. (C) Large ears to aid in heat dispersion: In hot environments, animals need to dissipate excess body heat to avoid overheating. Large ears can help with this by increasing the surface area through which heat can be lost. This adaptation is also likely to be found in animals living in hot arid environments. (D) Short loops of Henle to maximize water secretion: Short loops of Henle would result in less water reabsorption and more water being excreted. In a hot arid environment where water is scarce, this would be a disadvantage because it would lead to greater water loss. Therefore, this feature is least likely to be found in animals living in such conditions. So the correct answer option is (D) Short loops of Henle to maximize water secretion

Question: A survey of 1000 registered voters revealed that 450 people would vote for candidate A in an upcoming election. If 220,000 people vote in the election, how many votes would the survey takers predict candidate A should receive?\n(A) 44,500\n(B) 48,900\n(C) 95,000\n(D) 99,000

Answer: To predict the number of votes candidate A should receive based on the survey, we can use a proportion. The survey indicates that 450 out of 1000 registered voters would vote for candidate A. We can set up the following proportion to find the predicted number of votes out of 220,000 voters: $$\frac{450}{1000} = \frac{x}{220,000}$$ To solve for x, we can cross-multiply and then divide: $$450 \times 220,000 = 1000 \times x$$ $$99,000,000 = 1000x$$ $$x = \frac{99,000,000}{1000}$$ $$x = 99,000$$ Therefore, the survey takers would predict that candidate A should receive $99,000$ votes. So the correct answer option is (D) 99,000

Question: A cannonball is fired horizontally at 10 m/s from a cliff. Its speed one second after being fired is about\n(A) 10 m/s\n(B) 14 m/s\n(C) 16 m/s\n(D) 20 m/s

Answer: To determine the speed of the cannonball one second after being fired horizontally, we need to consider both the horizontal and vertical components of its

motion. 1. Horizontal Motion: - The horizontal velocity (v_x) remains constant because there is no horizontal acceleration (assuming air resistance is negligible). - Given: v_x = 10 m/s 2. Vertical Motion: - The vertical velocity (v_y) changes due to the acceleration caused by gravity (g = 9.8 m/s^2). - Initially, the vertical velocity is zero because the cannonball is fired horizontally. - After one second, the vertical velocity can be calculated using the equation: v_y = g * t where t = 1 s. v_y = 9.8 m/s^2 * 1 s = 9.8 m/s 3. Resultant Speed: - The resultant speed (v) is found by combining the horizontal and vertical components using the Pythagorean theorem: v = sqrt(v_x^2 + v_y^2) v = sqrt((10 m/s)^2 + (9.8 m/s)^2) v = sqrt(100 + 96.04) v = sqrt(196.04) v \approx 14 m/s Therefore, the speed of the cannonball one second after being fired is about 14 m/s. So the correct answer option is (B) 14 m/s

Question: A 3-microfarad capacitor is connected in series with a 6-microfarad capacitor. When a 300-volt potential difference is applied across this combination, the total energy stored in the two capacitors is\n(A) 0.09 J\n(B) 0.18 J\n(C) 0.27 J\n(D) 0.41 J

Answer: To find the total energy stored in the two capacitors connected in series, we need to follow these steps: 1. Calculate the equivalent capacitance of the series combination: For capacitors in series, the reciprocal of the equivalent capacitance \(C_{\text{eq}} \) is the sum of the reciprocals of the individual capacitances: \[\frac{1}{C_{\text{eq}}} = \frac{1}{C_1} + \frac{1}{C_2} \] Given \(C_1 = 3 \, \mu\text{F} \) and \(C_2 = 6 \, \mu\text{F} \): \[\frac{1}{C_{\text{eq}}} = \frac{1}{3 \, \mu\text{F}} + \frac{1}{6 \, \mu\text{F}} = \frac{2}{6 \, \mu\text{F}} + \frac{1}{6 \, \mu\text{F}} = \frac{3}{6 \, \mu\text{F}} = \frac{1}{2 \, \mu\text{F}} \] Therefore, \[C_{\text{eq}} = 2 \, \mu\text{F} \] 2. Calculate the total energy stored in the equivalent capacitor: The energy \(E \) stored in a capacitor is given by: \[E = \frac{1}{2} C V^2 \] Here, \(C = C_{\text{eq}} = 2 \, \mu\text{F} = 2 \times 10^{-6} \, \text{F} \) and \(V = 300 \, \text{V} \): \[E = \frac{1}{2} \times 2 \times 10^{-6} \, \text{F} \times (300 \, \text{V})^2 \] \[E = \frac{1}{2} \times 2 \times 10^{-6} \times 90000 \] \[E = 1 \times 10^{-6} \times 90000 \] \[E = 0.09 \, \text{J} \] Therefore, the total energy stored in the two capacitors is \(\boxed{0.09 \, \text{J}} \). So the correct answer option is (A) 0.09

A.4 Instruction-Tuned Model Experiments

To investigate how our approach affects models of different training stages, we also include experiments show that our framework also benefits the LLaMA-3-8B-Instruct version in Figure 6 and Figure 7, which has undergone general instruction-tuning (Touvron et al., 2023a) to enhance performance in many aspects. Notably, we observe improvements on both the base and the instruction-tuned model versions, which suggests that our approach may generalize well even to well-trained models.

A.5 Commonsense and General Reasoning

We additionally study the performance of our method on commonsense and general reasoning tasks, specifically evaluating it on CommonsenseQA (Talmor et al., 2019), Winogrande (Sakaguchi et al., 2019), and the full MMLU (Hendrycks et al., 2021a) dataset as presented in Table 8. Despite the strong SFT baseline, which suffices for most questions requiring only one or two reasoning steps, we demonstrate that our method surpasses other preference optimization methods in terms of multi-step reasoning when the model is prompted to think step-by-step. Notably, on the Winogrande dataset, our framework achieves a significant improvement of 6.6% over the strongest preference optimization baseline, ORPO.

A.6 Evaluation of Reasoning Quality

To quantitatively measure the reasoning quality after training with different methods, we report the Inter. F1 metrics (Wang et al., 2023a) which compares the numerical objects that are consistent between the generated reasoning path and ground-truth reasoning path. We report the results for

Method	CSQA	Winogrande	MMLU
LLaMA-3-8B			
SFT	82.7	86.0	63.7
RFT	72.3	64.2	59.6
DPO	72.7	57.9	56.5
ORPO	76.8	67.0	59.7
Ours	79.7	73.6	62.0
LLaMA-3-8B-Instruct			
SFT	82.3	84.8	63.4
RFT	73.8	66.5	63.3
DPO	70.4	63.7	67.9
ORPO	74.0	67.7	63.9
Ours	77.7	71.2	65.6

Table 8: Additional evaluation results on commonsense and general reasoning tasks.

LLaMA-3-8B on GSM8K as shown in Table 7. The results demonstrate that our framework not only improves the final reasoning benchmark score, but also enhances the reasoning quality as measure by the Inter. F1 metric.