SELF-EXPLORE: Enhancing Mathematical Reasoning in Language Models with Fine-grained Rewards

Hyeonbin Hwang[1] **Doyoung Kim**[1] **Seungone Kim**[1,2] **Seonghyeon Ye**[1] **Minjoon Seo**[1]

KAIST AI[1] Carnegie Mellon University[2]
{hbin0701, doyoungkim, seonghyeon.ye, minjoon}@kaist.ac.kr seungone@cmu.edu

Abstract

Training on large amounts of rationales (*i.e.*, CoT Fine-tuning) has been found effective for improving mathematical reasoning of large language models (LLMs). However, acquiring human-authored solutions or augmenting rationales from proprietary models is costly and not scalable. In this paper, we study the problem of whether LLMs could *self-improve* mathematical reasoning capabilities. To this end, we propose SELF-EXPLORE, where the LLM is tasked to explore the first wrong step (*i.e.*, the first pit) within the rationale and use such signals as fine-grained rewards for further improvement. On the GSM8K and MATH test set, SELF-EXPLORE achieves 11.57% and 2.89% improvement on average across three LLMs compared to supervised fine-tuning (SFT). Our code is available here.

1 Introduction

Recent works have shown that large language models (LLMs) can solve complex reasoning tasks with Chain-of-Thought (CoT) Prompting, which involves generating a rationale before its final prediction (Wei et al., 2023; Kojima et al., 2023; OpenAI et al., 2023; Team et al., 2023). Such ability is especially evident for mathematical reasoning, where many times precise reasoning over multiple steps is required to reach the correct answer (Fu et al., 2023; Chen et al., 2023). Meanwhile, relatively smaller models have shown limited performance, and thus prior works have focused on augmenting rationales from proprietary LLMs and distilling them to smaller models (Li et al., 2022; Kim et al., 2023b; Mukherjee et al., 2023; Yu et al., 2023b; Liu et al., 2023a; Mitra et al., 2023, 2024a; Li et al., 2024).

However, acquiring high-quality solutions remains challenging. For humans, hand-crafting detailed step-by-step rationale annotations is time-consuming and costly (Kim et al., 2023a). On the other hand, using close-sourced models through APIs incurs high expenses and distillation-based methods are inherently limited by the performance of their teacher model, which acts as the upper bound (Gudibande et al., 2023; Ye et al., 2023). Hence, such strategies are limited in advancing frontier models (Stanton et al., 2021; Gudibande et al., 2023). One potential solution to address this issue is to improve general capability of LLMs through self-training (Gulcehre et al., 2023; Chen et al., 2024; Yuan et al., 2024).

Inspired by prior works that focus on aligning LLMs to user preferences through self-training, we propose SELF-EXPLORE, a training method designed to self-improve the **mathematical reasoning** capabilities of LLMs by extracting granular learning signals from its own generated rationales. Specifically, a target model conducts step-level exploration to identify the first wrong step (*i.e.*, first pit) within each rationale by sampling multiple continuations. Then, we construct a pair-wise dataset by sorting the rationales into positive and negative samples at a step level. Finally, by applying an arbitrary preference learning objective (*e.g.*, Direct Preference Optimization (DPO) (Rafailov et al., 2023)) on a step-level, we relatively increase the probability of generating positive rationales and lower the probability of generating negative ones in a fine-grained manner.

Through experiments, we find that SELF-EXPLORE constantly improves the performance across different base models (Mistral-7B (Jiang et al., 2023), Llemma-7B (Azerbayev et al., 2023), and Deepseek-Math 7B (Shao et al., 2024)) without any distillation from proprietary models. For each model, we observe a 13.19%, 10.23%, and 11.30% improvement on GSM8K (Cobbe et al., 2021) and a 1.98%, 3.16%, and 3.54% improvement on MATH (Hendrycks et al., 2021) compared to supervised fine-tuning (SFT). Also, we find that constructing a pair-wise dataset in a granular man-
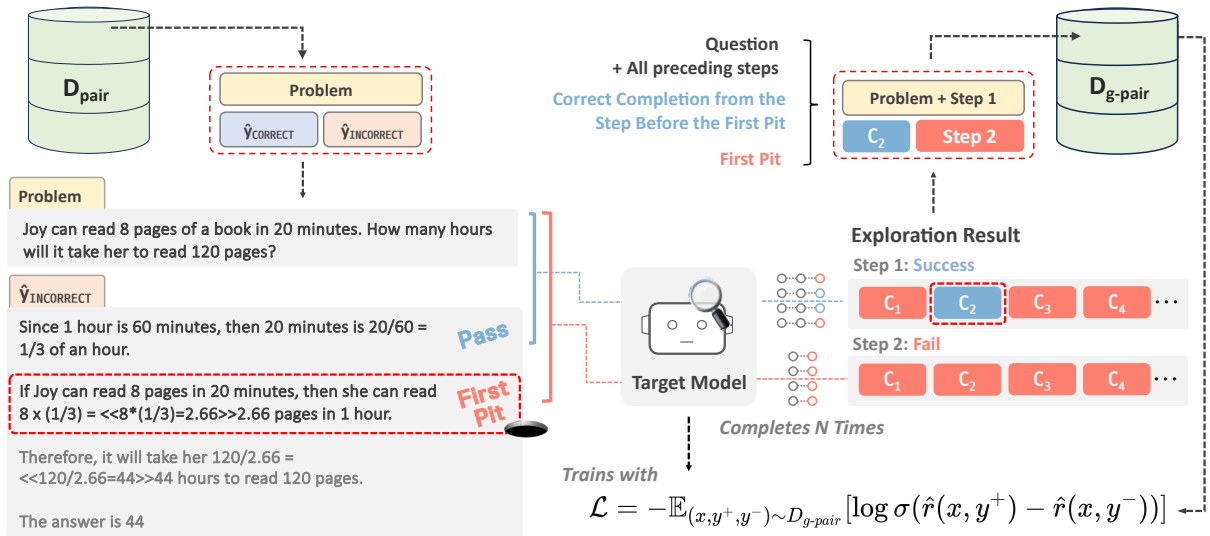
Figure 1: Overview of SELF-EXPLORE. From a pairwise dataset ($\mathcal{D}_{\text{pair}}$) made through outcome supervision, we use the incorrect rationales and make the target model generate multiple completions starting from each step. If none of the completions reach the answer, we mark that step as the *first pit*. Then, with the identified first pit, we reorganize $\mathcal{D}_{\text{pair}}$ into a granular preference dataset ($\mathcal{D}_{\text{g-pair}}$) which provides better learning signal during training.

ner based on a step-by-step basis (*i.e.*, identifying the first pit) outperforms a naive approach of constructing based on the correctness of the final prediction, leading to a 3.64% and 2.76% margin on the GSM8K and MATH dataset, respectively.

2 Related Works

2.1 Mathematical Reasoning of LLMs

To make a stronger math-reasoning model, previous works have either continually pre-trained the *base* model on large math corpus (Lewkowycz et al., 2022; Azerbayev et al., 2023) or used supervised fine-tuning with a large amount of synthetic dataset distilled from the frontier models (Luo et al., 2023; Yu et al., 2023b; Liu et al., 2023a; Mitra et al., 2024b; Shao et al., 2024; Toshniwal et al., 2024). There is also a growing number of works focusing on increasing test-time compute, namely generating multiple rationales then marginalizing over various reasoning paths (Wang et al., 2023), developing either an *outcome-level* or *process-level* separate verifier that could rank the rationales (Cobbe et al., 2021; Lightman et al., 2023; Liu et al., 2023a; Hosseini et al., 2024) or decoding under the guidance of a value-model (Xie et al., 2023; Liu et al., 2023b; Yu et al., 2023a). Our approach instead focuses on enhancing the model's top-1 performance which reduces test-time computational burden.

2.2 Step-level Supervision

Many studies have suggested the advantages of step-level guidance (Cobbe et al., 2021; Lightman et al., 2023), yet acquiring such labels is expensive. Thus, concurrent works rely on pseudo labels, evaluating whether the model can reach the correct answer when provided up to each successive step as input (Wang et al., 2024a,b; Jiao et al., 2024; Havrilla et al., 2024b). However, most of these works leverage acquired labels to train a verifier model, which is either used for PPO (Schulman et al., 2017) or inference time re-ranking. Our approach does not require any separate module, much simplifying the overall framework.

2.3 Self-Training for Mathematical Reasoning

Another line of works focus on self-training methods that compensate for the scarcity of high-quality training data. This includes utilizing self-generated correct rationales for training (Zelikman et al., 2022; Huang et al., 2022; Yuan et al., 2023; Ni et al., 2023), and also self-generated incorrect rationales (Havrilla et al., 2024b; Hosseini et al., 2024) - which can together form a pairwise dataset that can be trained with preference learning techniques, such as Direct Preference Optimization (DPO) (Rafailov et al., 2023).

These strategies are particularly effective for many Math Word Problem (MWP) tasks, where models demonstrate a much higher performance

when multiple attempts are allowed (pass@k) rather than just one (pass@1) (Havrilla et al., 2024a). This indicates that the model indeed has the potential to reach the correct answer, yet its answer distribution is misaligned. Our work aims to more precisely steer this distribution towards more optimal policy with fine-grained supervision.

3 Preliminaries

Given a language model π_θ and a dataset $\mathcal{D}$, *Self-Training* algorithms comprise of two stages: (1) dataset growth, where the dataset $\mathcal{D}$ is augmented with a π_θ's generations, and (2) policy improvement, where the pre-trained model improves human-alignment through preference learning followed by supervised fine-tuning (Gulcehre et al., 2023; Yuan et al., 2024; Chen et al., 2024). Here, we describe two relevant methods that are employed in our framework for self-training.

3.1 Rejection Sampling Fine-Tuning

RFT (Yuan et al., 2023) is a training method where the pre-trained model M_{PT} is allowed to fine-tune on its own correct generations. To do so, we first need a base generator with zero-shot reasoning ability, which is obtained by training M_{PT} on the initial dataset $\mathcal{D}$ with the MLE objective:

$$\mathcal{L}_{\text{MLE}} = -\sum_{i=1}^{|\mathcal{D}|} \log p_\theta(y_i | x_i) \qquad (1)$$

With the resulting model M_{SFT}, we sample N candidate rationales $\hat{y}_i$ for each question with a nonzero temperature T to form $\mathcal{D}_{\text{GEN}} = \{(x_i, \hat{y}_{i,j})_{j=1}^N \mid x_i \in Q\}$. After removing duplicate rationales using heuristics, each solution $\hat{y}_{i,j}$ is labeled as correct or incorrect by extracting their predicted final answer with extractor function $\mathcal{F}$ and comparing to the actual answer a_i. This set of correct rationales forms $\mathcal{D}_{\text{RFT}}$, and M_{PT} is trained on this dataset with objective in eq. 1. We highlight that in domains with a sufficiently large answer space, (*i.e.* numeric), a correct final answer strongly indicates that the rationale is likely error-free, which is a notable advantage of mathematical reasoning tasks.[1]

---

[1] In contrast, if the answer space is small (*i.e.* true/false or multiple choice) selecting the correct option does not necessarily guarantee that the rationale is also correct.
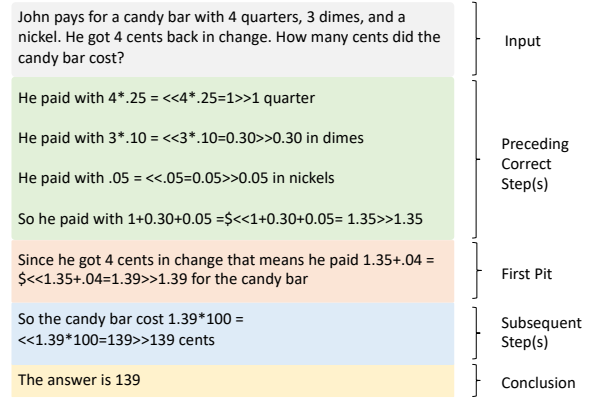

Figure 2: Example of a rejected sample from GSM8K: In the First Pit, 0.04 was mistakenly added instead of being subtracted.

3.2 Direct Preference Optimization

DPO (Rafailov et al., 2023) training requires a pairwise dataset consisting of a chosen completion y^+ and a rejected completion y^- for a given input x. Its objective relatively increases the log-likelihood of the chosen completion over the rejected one:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}[\log \sigma(\hat{r}_\theta(x, y^+) - \hat{r}_\theta(x, y^-))]$$

$$\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \qquad (2)$$

Here, reference model π_{ref} is generally initialized with supervised fine-tuning (SFT) with preferred completions for a single epoch to minimize distribution shift from the true reference distribution.

4 Method

4.1 Self-Training with Outcome Supervision

To achieve autonomous improvement for multi-step reasoning, we follow the general offline preference recipe from recent models (SFT + DPO) (Tunstall et al., 2023; Ivison et al., 2023). Yet we only utilize the initial human-curated dataset $\mathcal{D}$ and the training model's self-generated data. In this light, we initialize the reference policy π_{ref} by applying Rejection Sampling Fine-Tuning to the pre-trained model M_{PT} to obtain M_{RFT}.

To construct the pairwise dataset $\mathcal{D}_{\text{pair}}$, we start by adopting the conventional approach of designating a correct solution as a favorable sample (y^+) and an incorrect solution as an unfavorable sample (y^-) for a given problem x, using outcome supervision to determine correctness (Yu et al., 2023a; Hosseini et al., 2024). We pair each correct solution $\hat{y}_{i,j}$ in $\mathcal{D}_{\text{RFT}}$ with the incorrect solution $\hat{y}_{i,k}$

from $\mathcal{D}_{\text{GEN}}$ that has maximum edit distance in-between, in light of Pal et al. (2024). Overall, we utilize each solution only once and continue this pairing process until no further pairs can be formed. For additional details on pair formation, please see Appendix C.

After forming the pairwise dataset $\mathcal{D}_{\text{pair}}$, we train the model M_{RFT} using the objective specified in eq. 2. This approach guides the model on a *holistic* level to favor policies that generate solutions leading to the correct answer, relative to those that result in an incorrect answer. In the following sections, references to DPO specifically denote this outcome-supervised preference learning approach which we employ as a baseline method for our experiments.

4.2 Multi-Step Preference Learning

In preference learning, a language model π_θ functions as an agent optimized to generate responses that maximize a given reward function. In a multi-step problem setting, given an input x and a target sequence y comprising steps $\{y^1, ..., y^n\}$, we can define the agent's reward by evaluating the predicted final answer at the terminal state y^n against the ground truth answer a using the extractor function $\mathcal{F}$.

$$r(x, (y^1, ..., y^n)) = \begin{cases} 1, & \text{if } \mathcal{F}(y^n) = a \\ -1, & \text{if } \mathcal{F}(y^n) \neq a \end{cases} \quad (3)$$

If the answer space is large enough, we can safely assume that a match between these two indicates that all the prior steps $\{y^1, ..., y^{n-1}\}$ are also correct. On the other hand, if the terminal state y_n reached an incorrect answer, this suggests that the sequence generation has encountered at least one "*pit*" - an irreversible error in its prior steps that caused the agent to deviate from the correct path. Yet once we identify the *first pit*, we can consider all subsequent steps as non-relevant, given that they are already compromised by the preceding pit. Then, we can re-design the reward for generating each step y^i in the multi-step problem setting as follows:

$$r(x, (y^1, ..., y^i)) = \begin{cases} -1, & \text{if } y^i \text{ is a } \textit{first pit} \\ 1, & \text{if } i = n \\ & \text{and } \mathcal{F}(y^i) = a \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Meanwhile, the challenge posed by multi-step tasks is that it is hard to avoid the pit and reach the correct terminal state, especially when the problem requires many steps to solve. For simplicity, if we assume there is a constant probability of ϵ to fall into the pit in each stage, then the expected reward after generating t steps becomes $(1 - \epsilon)^t$, which exponentially decreases as t gets larger. In order to minimize this *risk*, previous works have utilized DPO to enable the original model as a reward model, steering away from the episodes that the model fell into the *pit*. However, DPO objective shown in eq. 2 relatively decreases the likelihood of all tokens in the rejected solution y^-. In light of eq. 4, we claim that only the step corresponding to the *first pit* should be discouraged. To elucidate, we consider the following two cases.

(1) Steps before the first pit For a rejected solution $y^- = \{y^1, ..., y^n\}$, there always exists an initial wrong step y_{w} corresponding to the *first pit*. If $w \neq 1$, the reward of the preceding steps $r(y^i|x, y^1, ..., y^{i-1})$, such that $i \leq w - 1$ should not be penalized.

(2) Steps after the first pit For the steps subsequent to y_w, while it's clear that y^w is flawed, decreasing the likelihood of $\sum_{i=w+1}^{n} P(y^i|x, y^1, ..., y^{i-1})$ could adversely impact the coherency of the model. This concern arises because the error in y^w may be due to a minor computation error or wrong formula construction, whereas the subsequent reasoning and steps could still be logically sound. (Figure 2)

4.3 Self-Explore

In this light, we apply the reward design from eq. 4 to transform $\mathcal{D}_{\text{pair}}$ into step-level **granular** pairwise dataset $\mathcal{D}_{\text{g-pair}}$. This requires modifying each *rejected* sample within $\mathcal{D}_{\text{pair}}$ so that we only reduce the likelihood of the **first pit** y^w. To find such a step, we employ our model as a self-guided explorer.

We assess whether the target model can reach the correct answer by sampling k completions with a non-zero temperature T from each step. If none of the completions yield the correct answer, we label that step as y^w. This indicates that the step has low *Q-value* or potential, suggesting that the step is either incorrect or is beyond the model's capability to utilize it effectively. On the other, if we do not find y^w until the end, we discard that sample. This is because the absence of y^w suggests
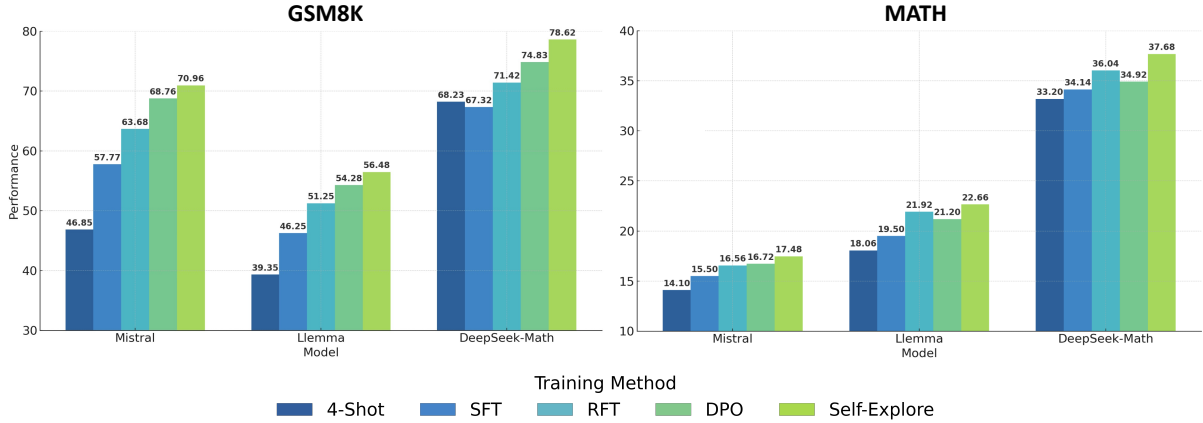
Figure 3: Result of three models trained with diverse learning methods. SELF-EXPLORE shows consistent superiority over other training methods in both GSM8K and MATH benchmark. For 4-Shot, we report the best performance achieved across three distinct prompts.

that the sample, produced by the base generator (M_{SFT}), may not actually be *infeasible* from the perspective of the explorer (M_{RFT}).

To form a $\mathcal{D}_{\text{g-pair}}$ instance, we set the first pit s^w as the new rejected sample. The new input is then created by concatenating the original input (question) with all steps prior to the first pit. For the new chosen sample, we randomly select one correct completion from the step just before the first pit (s^{w-1}), that matches this new input. We intentionally use the *whole* completion from the explorer to maximize the expected learning signal, thus the likelihood of deriving the correct answer. In a similar manner, if $w = 1$, we simply use the original chosen sample. Finally, we train with preference learning objective in eq. 2 on our reference model M_{RFT} using this new dataset $\mathcal{D}_{\text{g-pair}}$.

Our step-level annotation strategy builds on the framework first introduced in Wang et al. (2024a). However, unlike Wang's approach which utilizes different models for each role (*i.e.* completer, target model, and reward model), our method forms a preference pair using this label which allows the integration of these distinct systems into a single model, much simplifying the overall training process.

4.4 Experiments

Datasets and Models We conduct our experiments on two widely used MWP datasets GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). GSM8K dataset consists of 7,473 training and 1,319 test problems, while the MATH dataset contains 7,500 training and 5,000 test problems.

We test SELF-EXPLORE across 3 different models: Mistral-7B (Jiang et al., 2023), Llemma-7B (Azerbayev et al., 2023), and Deepseek-Math-7B-Base (Shao et al., 2024).[2]

Hyperparameters For the base generator M_{SFT}, we only train for 2 epochs, yet report the performance of the best checkpoint over 5 training epochs to ensure a fair comparison. Similarly, For M_{RFT}, we train the model for one epoch, yet report the best performance achieved over the course of 5 epochs. For all supervised fine-tuning, we use overall batch size of 64 and conduct learning rate search between $\{1e^{-6}, 1e^{-5}\}$ for all models. To construct $\mathcal{D}_{\text{RFT}}$, we use $N = 100$, with $T = 0.7$. For step-level exploration, we also use temperature of 0.7, and generate $k = 4$ at each step. All our generations were carried out using *vllm* (Kwon et al., 2023). For DPO training, we use overall batch size of 32, conduct learning rate search among $\{1e^{-6}, 5e^{-6}, 1e^{-7}\}$, and train for 3 epochs to report the best performance.

5 Results

5.1 Main Results

As shown in Figure 3, SELF-EXPLORE shows the highest performance in MATH and GSM8K compared to other methods. Especially, our method shows 13.19%, 10.23%, 11.30% increase in GSM8K and 1.98%, 3.16%, 3.54% increase in MATH compared to Supervised Fine-Tuning (SFT)

---

[2]All datasets and models are under MIT license, except for Mistral-7B which is under Apache 2.0. We use these solely for research purposes.

Data Type	GSM8K	MATH
Pairwise	74.83	34.92
Granular Pairwise	**78.47**	**37.68**
- Choose only First Step	75.74	35.76
- Reject All	75.89	36.82

Table 1: DeepSeek-Math's GSM8K test set accuracy when trained with DPO on various types of preference data.

for each model. Also, it consistently performs better than training DPO with outcome-supervised rewards from $\mathcal{D}_{\text{pair}}$, which shows the strength of our step-level reward.

Meanwhile, DPO performs worse than RFT in MATH dataset for Llemma and DeepSeek-Math. Note that this does not mean that DPO brought performance degradation, but rather RFT (1 epoch) + DPO achieved less performance than the optimal checkpoint achieved by RFT alone. For instance, when DPO was applied to the one-epoch RFT checkpoint, the performance showed a marginal increase from 34.82 to 34.92, whereas applying SELF-EXPLORE to the same checkpoint achieved 37.68. Unlike the granular supervision provided by SELF-EXPLORE, we hypothesize that outcome supervision offers a significantly weaker training signal. This weaker signal is more challenging for the model to interpret and utilize effectively, making it harder to guide the model towards a successful policy. This may rather lead to reward exploitation or undesired penalization of correct steps that may not necessarily improve its general reasoning ability.

We also note that the performance gain in MATH is much lower when compared to GSM8K, which is primarily due to its difficulty. Not only the task itself is more inherently challenging, but also the training dataset is limited in size, which is then tested against a large pool of test problems. We hypothesize that low performance of M_{RFT} as both generator and completer prevents an effective exploration process when conducting both overall generation and step-level search. In fact, for the MATH dataset, we observe number of unique question-level samples in $\mathcal{D}_{\text{RFT}}$ resulting significantly less. For more details about the dataset statistics, please refer to Appendix D.

5.2 Step-Level Reward Design

To better justify our design for step-level fine-grained reward, we conducted tests on DeepSeek-Math using two additional settings from our current

dataset, $\mathcal{D}_{\text{g-pair}}$. 1) Choose Only First Step: For the new chosen sample, we take only the *first correct step*, rather than the *entire* completion. This approach aligns with the new rejected sample, where we only minimize the likelihood of the first *pit* alone. 2) Reject All: For the new rejected sample, we reject the first pit along with its all subsequent steps. We no longer regard the steps after the first pit as irrelevant; instead, we aim to reduce their likelihood as well.

As shown in Table 1, we observe that training with our fine-grained reward yields the best performance in both datasets. While the two other settings perform better than training with outcome-supervised pairwise dataset, they both result in sub-optimal performances. This again highlights the idea that the learning signal becomes the most effective when maximally utilizing the whole correct solution while decreasing only the first pit, which is in line with the eq. 4.

Dataset	$k = 4$	$k = 8$	$k = 16$	$k = 32$
GSM8K	**70.96**	69.9	70.81	70.05
MATH	**17.48**	17.4	17.44	17.10

Table 2: Performance of Mistral-7B for GSM8K and MATH datasets, with varying exploration size k.

6 Analysis

6.1 Ablation Studies

Effect of Exploration Space We further analyze whether larger exploration space leads to a better performance. Specifically, we aim to analyze whether steps in the rejected sample which have **low, non-zero** total expected reward (*i.e.* low probability of reaching to the correct answer) should not be discouraged. These could be found by exploring more paths with larger k. On the other, one could argue that it is better to prevent the model from going through such path from the outset by rigorously evaluating each step against a strict standard of smaller k. Therefore, we test Mistral-7B with varying step-level exploration size k among $\{4, 8, 16, 32\}$, with which we accordingly build each $\mathcal{D}_{\text{g-pair}}$ and train the target model with the DPO objective.

As shown in Table 2, we see that increasing exploration size does not lead to performance increase, yet rather often leads to degradation. First pit detection indeed does occur in later stages when using larger exploration space - for instance, for

Method	Acc.
RFT	63.68
DPO	66.64
SELF-EXPLORE: Completers	
$\text{Mistral}_{\text{SFT}}$	67.70
$\text{Mistral}_{\text{RFT}}$ (Ours)	68.46
$\text{DeepSeek}_{\text{RFT}}$	66.79
GPT-4	**69.14**

Table 3: GSM8K Test Set Accuracy of the Mistral-7B when trained DPO with 5.8K instances of supervised by different completers.

MATH dataset the mean index of s^w becomes $1.86 \rightarrow 2.19 \rightarrow 2.61 \rightarrow 3.13$ with increasing k values. However, this does not necessarily extend to a better resulting model performance.

We believe that while it may be technically *feasible* to reach an answer through a certain step, it does not necessarily mean that it is *favorable*. For instance, if a model has a high probability ϵ of falling into the pit after a given correct step (*i.e.* it tends to associate post-sequences that is logically incorrect), sometimes it may be more effective to avoid such step from the beginning, if there are other correct alternatives that can lead to the correct answer with less future *risk*. In this manner, we hypothesize it is favorable to optimize the steps with high total expected rewards, or otherwise it may introduce unnecessary noise.

Effect of Explorer We also investigate the potential of enhancing model performance by adopting a different explorer (or supervisor). Current labeling method guarantees a fairly reasonable step-level accuracy (Wang et al., 2024a), yet as $\mathcal{D}_{\text{g-pair}}$ data quality heavily depends on the explorer's capability, we hypothesize that our final model performance may be bottlenecked by the explorer's limitations.

With this end, we train DPO objective on Mistral-7B M_{RFT} with $\mathcal{D}_{\text{g-pair}}$ completed by a range of models, *i.e.* $\text{Mistral}_{\text{SFT}}$, $\text{Mistral}_{\text{RFT}}$, $\text{Deepseek-Math}_{\text{RFT}}$, and GPT-4 (OpenAI et al., 2023). We use the same step-level exploration approach in SELF-EXPLORE except for GPT-4, which showed tendency to identify the wrong step instead of completing from the given steps even when provided with explicit instructions. Therefore, we directly prompted GPT-4 to pinpoint the first wrong step and to generate correct sequence from there while ensuring it maintains the original style of the preceding steps. To leverage GPT-4

as the oracle completer, we curated a specialized subset of $\mathcal{D}_{\text{pair}}$ to start with. We first chose one sample per each unique problem $x_i \in \mathcal{D}_{\text{pair}}$, and only included samples where GPT-4 successfully arrived at the correct conclusion, resulting in total of 5.8K samples.

As shown in Table 3, we see applying DPO with either $\mathcal{D}_{\text{pair}}$ and $\mathcal{D}_{\text{g-pair}}$ results in lower performance due to the dataset's smaller size. Yet, we observe that SELF-EXPLORE still performs better than outcome-supervised DPO in small-scale. Also, while $\text{DeepSeek}_{\text{RFT}}$ itself performs better as a *generator* than $\text{Mistral}_{\text{RFT}}$ (*i.e.* 71.42 vs 63.68), as a completer for $\text{Mistral}_{\text{RFT}}$, the former yields higher efficiency. We deduce this may be due to the fact that DPO generally works better when the training data, especially when the chosen completions are closer to its distribution, which is also suggested by the common practice of training SFT for one epoch prior to DPO (Rafailov et al., 2023; Yuan et al., 2024).

Finally, we observe that using oracle completer GPT-4 results in a better final model performance than using the same model's M_{RFT}. We believe that as the generated completions by GPT-4 does not fully represent the target model's distribution, if the completions were generated by a hypothetical *oracle* M_{RFT} of the same model, performance would have been even higher. We believe that this suggests that our method could be further improved with more robust exploration methods.

Effect of Objective Function We also analyze whether the effectiveness of our fine-grained data can be extended to other preference learning objectives, such as IPO (Azar et al., 2023) and KTO (Ethayarajh et al., 2024). With other settings equal, we train Mistral-7B's M_{RFT} using $\mathcal{D}_{\text{pair}}$ and $\mathcal{D}_{\text{g-pair}}$, for 1 epoch and $\tau = 0.01$ for IPO.

In Figure 4, we see that for both datasets using fine-grained supervision consistently results in better model performance than using outcome-supervised pairwise data. This shows the robustness of SELF-EXPLORE across various objectives, highlighting the general effectiveness of our fine-grained data. We have also experimented using high values of τ for IPO and ORPO (Hong et al., 2024), however they showed degraded performance for both types of supervisions.[3]

---

[3]We posit that the efficacy of self-training hinges on the introduction of a strong distinct positive signal for the chosen examples and negative signal for the rejected ones.
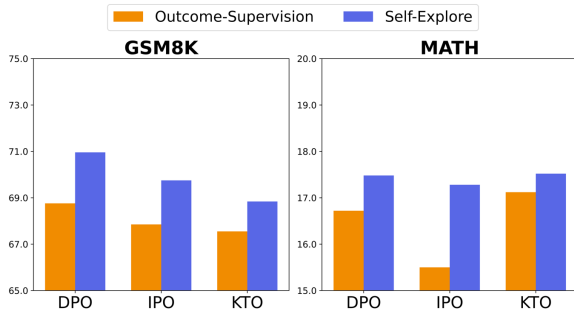
Figure 4: Mistral-7B performance when trained with different preference learning objectives using outcome-level supervision ($\mathcal{D}_{\text{pair}}$) or SELF-EXPLORE ($\mathcal{D}_{\text{g-pair}}$).

6.2 Qualitative Analysis

We also qualitatively analyze whether the numerical performance gains also translate into improved solution quality. To do so, we randomly select 100 questions from GSM8K[4] Test set and generate response from DeepSeek-Math models trained with RFT, DPO, and SELF-EXPLORE. Then, we use GPT-4 as our evaluator using FLASK (Ye et al., 2023), effectively assessing the given solution's logical robustness, efficiency and correctness in a scale of 1-5 against the ground truth solution.

As shown in the Table 4, we see that SELF-EXPLORE scores the best result in all criteria. Also, the general trend in the table implies that increased numerical performance does indicate a better quality in terms of correctness, robustness, and efficiency. We hypothesize that our method guides the model to better utilize its available knowledge, leading to the generation of solutions that are both more efficient and robust. For additional details and examples on FLASK evaluation, please see Appendix G.

7 Conclusion

In this paper, we propose SELF-EXPLORE where LLMs can self-improve from given initial human curated dataset using fine-grained supervision. By utilizing automatic self-exploratory annotation, SELF-EXPLORE effectively integrates the roles of the annotator, target, and reward models into a single system. On mathematical reasoning datasets GSM8K and MATH, our method outperforms traditional supervised fine-tuning (SFT) method by 11.57% and 2.89% in average across three different models, respectively. Furthermore, we demonstrate

---

[4]We use GSM8K to guarantee a robust evaluation performance of GPT-4.

that our method introduces minimal computation overhead (See Appendix H). We hope our work could motivate future works to explore self-training methods that could more robustly generalize to a broader reasoning space across various domains, with ends of advancing the frontier of LLM reasoning.

Model	Robustness	Correctness	Efficiency
RFT	3.87	3.86	4.07
DPO	4.19	4.15	4.35
Self-Explore	**4.27**	**4.28**	**4.44**

Table 4: Comparison of FLASK Logical Metrics Across Different Training Methods, using DeepSeek-Math on GSM8K.

Limitations

We propose a method on how to better exploit the *solution* space to provide a better fine-grained supervision for self-improving reasoning capabilities. Yet given limited amount of questions, which is a quite common scenario, preference learning with self-generated samples may be prone to overconfidence and thus increases top-1 performance at the expense of diminished test-time exploration robustness (Cao et al., 2024). We suspect this is related to reward overoptimization (Gao et al., 2022; Burns et al., 2023) and attach relevant analysis in Appendix. We leave as a future work on methods for mitigating this overoptimization, where one promising direction could be exploring the potential of integrating collection of diverse datasets as in Longpre et al. (2023), so that the model can generalize across a broader question space.

Also, our work is currently conducted with 7B pre-trained models and does not consider extensively fine-tuned CoT models or larger scale architectures that have shown stronger reasoning capabilities (Yu et al., 2023b; Mitra et al., 2024b; Shao et al., 2024). We believe for practical self-training applications, it is crucial to explore continual training processes on these sophisticated models. While this paper aims to compensate for distilled rationales used in instruction-tuned models, we encourage future works to investigate about such models could further benefit from self-improvement processes in a robust and effective manner.

Acknowledgements

References

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. *arXiv 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. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Boxi Cao, Keming Lu, Xinyu Lu, Jiawei Chen, Mengjie Ren, Hao Xiang, Peilin Liu, Yaojie Lu, Ben He, Xianpei Han, Le Sun, Hongyu Lin, and Bowen Yu. 2024. Towards scalable automated alignment of llms: A survey.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*.

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling laws for reward model overoptimization.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. 2024a. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Railneau. 2024b. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Seungone Kim, Se June Joo, Yul Jang, Hyungjoo Chae, and Jinyoung Yeo. 2023a. Cotever: Chain of thought prompting annotation toolkit for explanation verification. *arXiv preprint arXiv:2303.03628*.

Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023b. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. *arXiv preprint arXiv:2305.14045*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. 2024. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*.

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

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023a. Tinygsm: achieving >80% on gsm8k with small language models. *arXiv preprint arXiv:2312.09241*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023b. Don't throw away your value model! making ppo even better via value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024a. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024b. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.

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

Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Oleksandr Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. 2023. Learning math reasoning from self-sampled correct and partially-correct solutions. *arXiv preprint arXiv:2205.14318*.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin,

Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. 2021. Does knowledge distillation really work? *arXiv preprint arXiv:2106.05945*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman

Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Martin Chadwick, Gaurav Singh Tomar, Xavier Garcia, Evan Senter, Emanuel Taropa, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Yujing Zhang, Ravi Addanki, Antoine Miech, Annie Louis, Laurent El Shafey, Denis Teplyashin, Geoff Brown, Elliot Catt, Nithya Attaluri, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaly Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, Hanzhao Lin, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin

Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yong Cheng, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, YaGuang Li, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević,

1455

Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Gamaleldin Elsayed, Ed Chi, Mahdis Mahdieh, Ian Tenney, Nan Hua, Ivan Petrychenko, Patrick Kane, Dylan Scandinaro, Rishub Jain, Jonathan Uesato, Romina Datta, Adam Sadovsky, Oskar Bunyan, Dominik Rabiej, Shimu Wu, John Zhang, Gautam Vasudevan, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Betty Chan, Pam G Rabinovitch, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Sahitya Potluri, Jane Park, Elnaz Davoodi, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Chris Gorgolewski, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Paul Suganthan, Evan Palmer, Geoffrey Irving, Edward Loper, Manaal Faruqui, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Michael Fink, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marin Georgiev, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Minnie Lui, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Lam Nguyen Thiet, Daniel Andor, Pedro Valenzuela, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Sarmishta Velury, Sebastian Krause, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Tejasi Latkar, Mingyang Zhang, Quoc Le, Elena Allica Abellan, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Sid Lall, Ken Franko, Egor Filonov, Anna Bulanova, Rémi Leblond, Vikas Yadav, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Hao Zhou, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Jeremiah Liu, Mark Omernick, Colton Bishop, Chintu Kumar, Rachel Sterneck, Ryan Foley, Rohan Jain, Swaroop Mishra, Jiawei Xia, Taylor Bos, Geoffrey Cideron, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Petru Gurita, Hila Noga, Premal Shah, Daniel J. Mankowitz, Alex Polozov, Nate Kushman, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Anhad Mohananey, Matthieu Geist, Sidharth Mudgal, Sertan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Quan Yuan, Sumit Bagri, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Aliaksei Severyn, Jonathan Lai, Kathy Wu, Heng-Tze Cheng, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Mark Geller, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Andrei Sozanschi, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Abhimanyu Goyal, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Sabaer Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Tao Zhu, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Dustin Tran, Yeqing Li, Nir Levine, Ariel Stolovich, Norbert Kalb, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Balaji Lakshminarayanan, Charlie Deck, Shyam Upadhyay, Hyo Lee, Mike Dusenberry, Zonglin Li, Xuezhi Wang, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Summer Yue, Sho Arora, Eric Malmi, Daniil Mirylenka, Qijun Tan, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Steven Zheng, Francesco Pongetti, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Ragha Kotikalapudi, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Chenkai Kuang, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Pei Sun, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Ishita Dasgupta, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Yuan Liu, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Ivo Penchev, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Adam Kurzrok, Lynette Webb, Sahil Dua, Dong Li, Preethi Lahoti, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang

1456

Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Taylan Bilal, Evgenii Eltyshev, Daniel Balle, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Adams Yu, Christof Angermueller, Xiaowei Li, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Kevin Brooks, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Komal Jalan, Dinghua Li, Ginger Perng, Blake Hechtman, Parker Schuh, Milad Nasr, Mia Chen, Kieran Milan, Vladimir Mikulik, Trevor Strohman, Juliana Franco, Tim Green, Demis Hassabis, Koray Kavukcuoglu, Jeffrey Dean, and Oriol Vinyals. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

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. *arXiv preprint arXiv:2312.08935*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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. *arXiv preprint arXiv:2402.02658*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. *arXiv preprint arXiv:2305.00633*.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. Flask: Fine-grained language model evaluation based on alignment skill sets. *arXiv preprint arXiv:2307.10928*.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023a. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

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

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*.

A Post-Training Distribution

Here we analyze how the model's distribution changes after applying different training methods, including RFT, DPO and SELF-EXPLORE. Specifically, we use our best performing model DeepSeek-Math with a special focus on MATH dataset to explore potential directions on how LLMs could better self-improve in more advanced reasoning capabilities.

While we previously used greedy decoding to report the top-1 performance, here we sample 100 predictions per problem from the test set with temperature of 0.7 and sort the generations by the overall sequence likelihood in descending order. Then, we report its performance in three different metrics, total accuracy, self-consistency (maj@k), and pass@k, in Figure 5.

For the total accuracy, we observe a general trend of curves decreasing with the inclusion of samples with lower overall likelihood. Yet, we observe DPO and SELF-EXPLORE displaying smaller gaps of reduction. Numerically, as k goes from 1 to 100, SELF-EXPLORE performs $0.388 \rightarrow 0.367$, DPO $0.367 \rightarrow 0.337$, and RFT $0.376 \rightarrow 0.336$. We believe preference learning with self-generated samples minimizes the *risk* as even token generation with comparatively lower likelihood to be sampled eventually lead to the correct answer.

However, this comes at the cost of reduction in sample diversity where preference learning (*i.e.* RLHF) has been previously reported to promote similar phenomena (Kirk et al., 2024). We believe this even intensifies as we are training with our own generated data. To support this, we leverage BERT (Devlin et al., 2019) to extract embeddings of model generations and express solution diversity as the average per-input pairwise distance of embeddings, *i.e.* for i^{th} sample, this is given as:

$$d_i = \frac{2}{N \cdot (N-1)} \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} d(h_{i,j}, h_{i,k})$$

where h is the embedding and $N = 100$ in this case.

We plot the distribution of d_i for each training method in the boxplot shown in Figure 6. We observe a general decrease in embedding distances from left to right. Particularly DPO and SELF-EXPLORE display lower embedding distance than SFT and RFT, hinting at relatively reduced diversity. This phenomena also explains why Pass@K

for SFT and RFT is higher compared to those trained with preference learning objective, as SFT and RFT may engage in more *exploration* during test-time.

In addition, it is important to recognize that a policy characterized by reduced diversity may exhibit limited generalization capabilities, which could be seen as a drawback. Note that for Self-Consistency (maj@k), RFT and SFT surpasses SELF-EXPLORE at $K = 6$ and 15, respectively. We find that the reason for this phenomena is due to the concentration of the answer space stemming from the lack of solution diversity, as demonstrated in Appendix B.

Our models trained with preference learning tend to heavily favor what they identify as an *optimal* answer. Specifically, the reward accuracy when training these models quickly converge to 1, which is illustrated in Appendix E, indicating a potential reward exploitation that may lead to limitation in the model's ability to generalize. We hypothesize that this stems from self-training focused on exploring a confined *solution* space, which may not effectively extend to a broader *question* space.

Consequently, the solution distribution becomes skewed, leading to the emergence of overly confident peaks (modes) that may accurately represent the training data but fail to generalize to new unseen questions during testing, as shown by the reduced diversity. In contrast, models trained with SFT or RFT adopt a more *uniform* distribution across potential answers, whereas marginalizing over answers allow for slightly more pronounced peaks to be observable. (*i.e.* Self-Consistency) Overall, these benefits appear diminished when training with preference learning objective with self-generated data.

In fact, we also observe a similar pattern for the solution distribution in GSM8K. There is also less reduction in total accuracy with increasing k for models trained with preference learning objective. This again can be explained as a *risk* minimization behavior. Regarding the other two metrics, we see that SFT and RFT models exhibit lower performance at lower k values, but they eventually converge to the similar level (maj@k) or even surpass (pass@k) with increasing k. We hypothesize this trend again reflects the reduction in diversity within the model's predictions for DPO and SELF-EXPLORE. At the same time, we see that for DPO, rather the performance increases with inclusion of lower-k predictions. This indicates a potential mis-
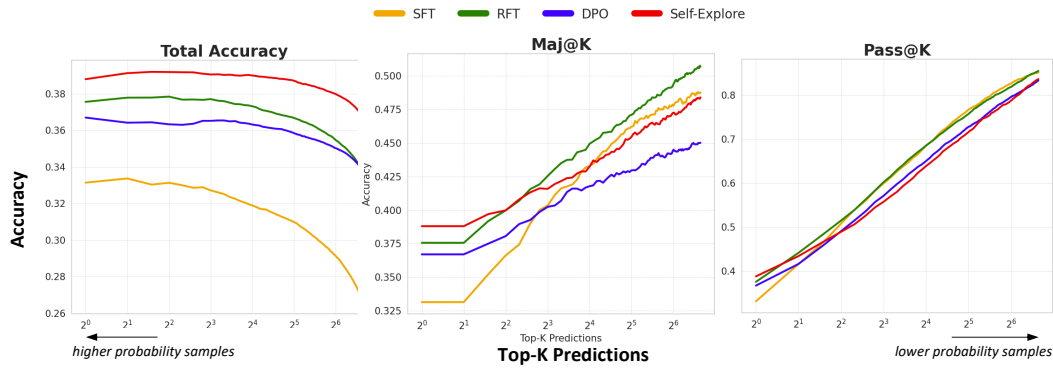
(a) Performance on **MATH**
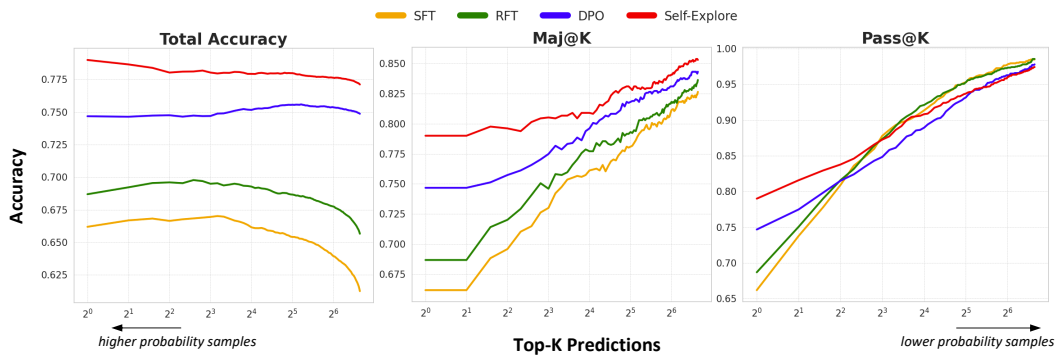
(b) Performance on **GSM8K**

Figure 5: Performance of DeepSeek-Math Model on different datasets when trained with diverse training methods - we report using three metrics: total accuracy, maj@k (*i.e.* self-consistency), and pass@k.

alignment, which explains the need for the granular supervision during training for a better learning signal.

B Answer Distribution

On the left side of Figure 7, we see that the number of unique answers decrease in order of SFT, RFT, DPO, then SELF-EXPLORE. Meanwhile on the right, we see that DPO and SELF-EXPLORE shows the highest proportion of dominant answer, suggesting a concentrated or skewed distribution of the answer space. These observations support the hypothesis that the model may exhibit over-confidence in its 'optimal' answers, when applied preference learning with self-generated solutions. Such confidence, without sufficient generalization power, could indicate potential overfitting to the training data.

C Pairwise Dataset Formation

C.1 Maximum Pair Constraints

We initially set no upper limit on the number of response pairs per problem in our dataset. However,
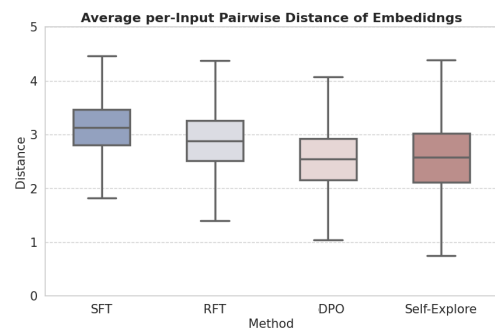

Figure 6: Average Per-Input Pairwise Distance of Embeddings of DeepSeek-Math, when trained with different methods.

preliminary analysis suggested that problems with nearly balanced correct and incorrect responses could potentially generate disproportionately many pairs, risking data overfitting. Thus, we have decided to adopt a maximum threshold of eight pairs (N=8) for each problem x_i. While we did not observe such cases many times, we adopted this strategy to ensure a more equitable distribution across different questions.
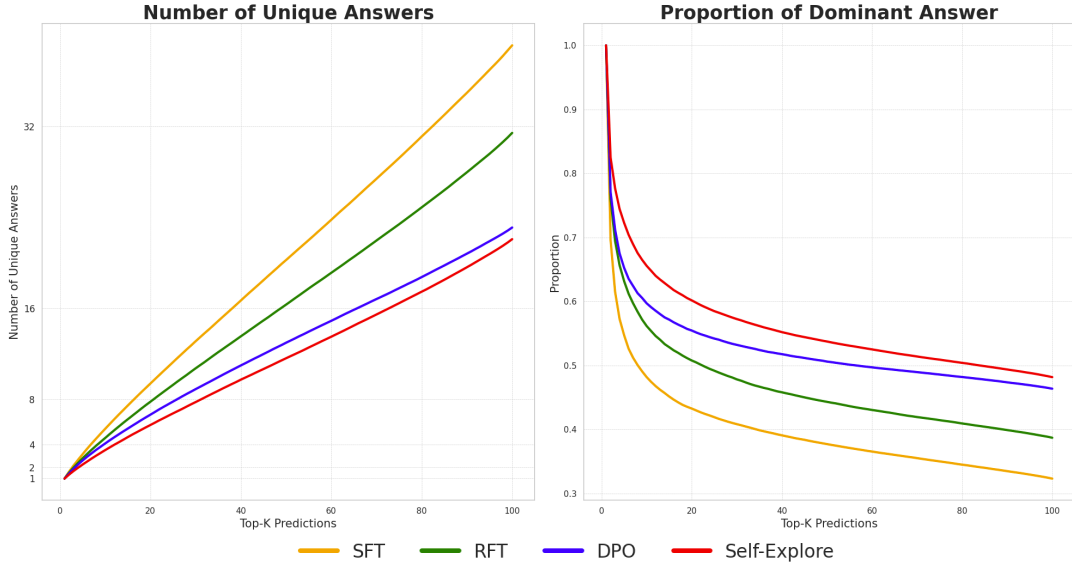
Figure 7: Final answer diversity and the proportion of the dominant (most common) answer within top-k predictions of DeepSeek-Math on MATH Test dataset.

C.2 Excluding Conclusion

When we first ran DPO training, we observed model performance significantly degrading when including *Conclusion* part within the rejected sample (Figure 2). In such case, our trained model was frequently presenting self-contradictory statements in the conclusion, yielding random answers that were unrelated to the reasoning presented in the preceding steps. We believe it is due to the concurrent presence of definitive statements like "The answer is X" in the chosen and "The answer is Y" in the rejected sample, causing model confusion during training. Therefore we decided to omit the conclusion section (+ *eos* token) from all **rejected** samples.

D Training Dataset Size

In this section, we discuss about the dataset size utilized for each training method and model. Despite the seemingly comparable amount of training samples (Table 5), we highlight several observations based on the proportion of *question-level* unique instances in each dataset, which is shown in Table 6 and 7:

1. Few *incorrect* samples for GSM8K Transitioning from the RFT to the paired dataset, there is a notable reduction in the number of unique questions for GSM8K compared to MATH. This occurred because in several instances, the model generated all 100 solutions correctly, or there were

fewer than four incorrect solutions. This overall hints at the scarcity of generated incorrect samples when training with GSM8K dataset.

2. Few *correct* samples for MATH Despite the model achieving high pass@k rates on the training set (over 90% for GSM8K and over 70% for MATH), the actual number of instances that *pass* is notably small for the MATH dataset. Especially, there is a large decline in Table 7 when considering the number of unique questions with more than 4 instances. This suggests that for many questions, the models barely reach the correct answer within 100 generations.

E DPO Training

While in the original paper (Rafailov et al., 2023), DPO training displayed chosen completion's win rate over the rejected completion around 60-70%, we observe in Figure 8 that the reward of *chosen* sample quickly suprasses that of *rejected* in our early stages, with winrate converging to 1 in both datasets. We hypothesize that this occurs for two reasons. 1) Chosen completion is generated by M_{RFT} which is closer to the target model's distribution, while rejected is generated by M_{SFT}. 2) Models can also quickly learn to distinguish the preference within the limited question numbers, which may nonetheless lead to overfitting.

F   Examples

In Figure 9, we see that while the DPO model concludes prematurely after the 5th step, falling into a *pit*, the SELF-EXPLORE model continues to generate subsequent steps robustly, ultimately arriving at the correct answer. This sample effectively illustrates how our method achieves step-level robustness through targeted step-level supervision.

G   FLASK Prompt

In Figure 10, we present the prompt used for the GPT-4 FLASK evaluation, which assesses three key logical skills: robustness, correctness, and efficiency. These skills are evaluated against the ground truth (GT) solution using a deterministic rubric for each criterion.

When evaluating the example responses present in Figure 9, we see that DPO model receives a score of $2, 3, 2$ while SELF-EXPLORE gets a full score of $5, 5, 5$ for Logical robustness, Correctness and Efficiency, respectively, as shown in Figure 11. GPT-4's coherent explanation of these scores adds credibility to the overall FLASK evaluation result in Table 4, underscoring the superior quality of responses generated by the SELF-EXPLORE model.

H   Computational Costs

We report the overall computational costs (I.E. GPU Hours) for each baseline, including the exploration stage in Table 8. Our baselines involve different training (Tr.) and generation (Gen.) stages. Note that the table reflects the following configuration:

- Mistral 7B

- 5 epoch SFT & RFT training

- 3 epoch DPO training

- 7.4M RFT samples generated

- 39K pairwise-data exploration

Methods	Mistral	Llemma	DeepSeek
Dataset: GSM8K			
FT	7,473	7,473	7,473
RFT	67,755*	38,989	52,005
pair	56,443*	37,058	38,872
g-pair	56,283*	36,812	38,618
Dataset: MATH			
FT	7,500	7,500	7,500
RFT	31,839	34,419	40,654
pair	31,527	34,124	39,769
g-pair	31,248	33,960	39,496

Table 5: Dataset size used for each training method, by each model.
* denotes no maximum pair formation constraint

	Mistral	Llemma	DeepSeek
FT	1.0	1.0	1.0
NUMBER OF SAMPLES ≥ 1			
RFT	0.9830	0.9252	0.9917
pair	0.9213	0.9113	0.8955
g-pair	0.9212	0.9098	0.8947
NUMBER OF SAMPLES ≥ 4			
RFT	0.7376	0.7281	0.8616
pair	0.6204	0.5739	0.6063
g-pair	0.6195	0.5700	0.6024

Table 6: Proportion of questions in GSM8K with at least N instances for each training method, by each model.

	Mistral	Llemma	DeepSeek
FT	1.0	1.0	1.0
NUMBER OF SAMPLES ≥ 1			
RFT	0.7345	0.7587	0.8240
pair	0.7345	0.7356	0.8225
g-pair	0.7345	0.7353	0.7971
NUMBER OF SAMPLES ≥ 4			
RFT	0.4904	0.5375	0.6479
pair	0.4844	0.5320	0.6309
g-pair	0.4819	0.5309	0.6292

Table 7: Proportion of questions in MATH with at least N instances for each training method, by each model.

Stages	SFT (Tr.)	RFT (Gen.)	RFT (Tr.)	Exploration (Gen.)	DPO (Tr.)	Total Time (hr)
GPU Hours	1.3	6	11.2	2.7	20	
SFT	✓					1.3
RFT	✓	✓	✓			18.5
DPO	✓	✓	✓		✓	38.5
Self-Explore	✓	✓	✓	✓	✓	41.2

Table 8: GPU hours for different baselines.

GSM8K: DPO

train/rewards/accuracies

GSM8K: Self-Explore

train/rewards/accuracies

MATH: DPO

train/rewards/accuracies

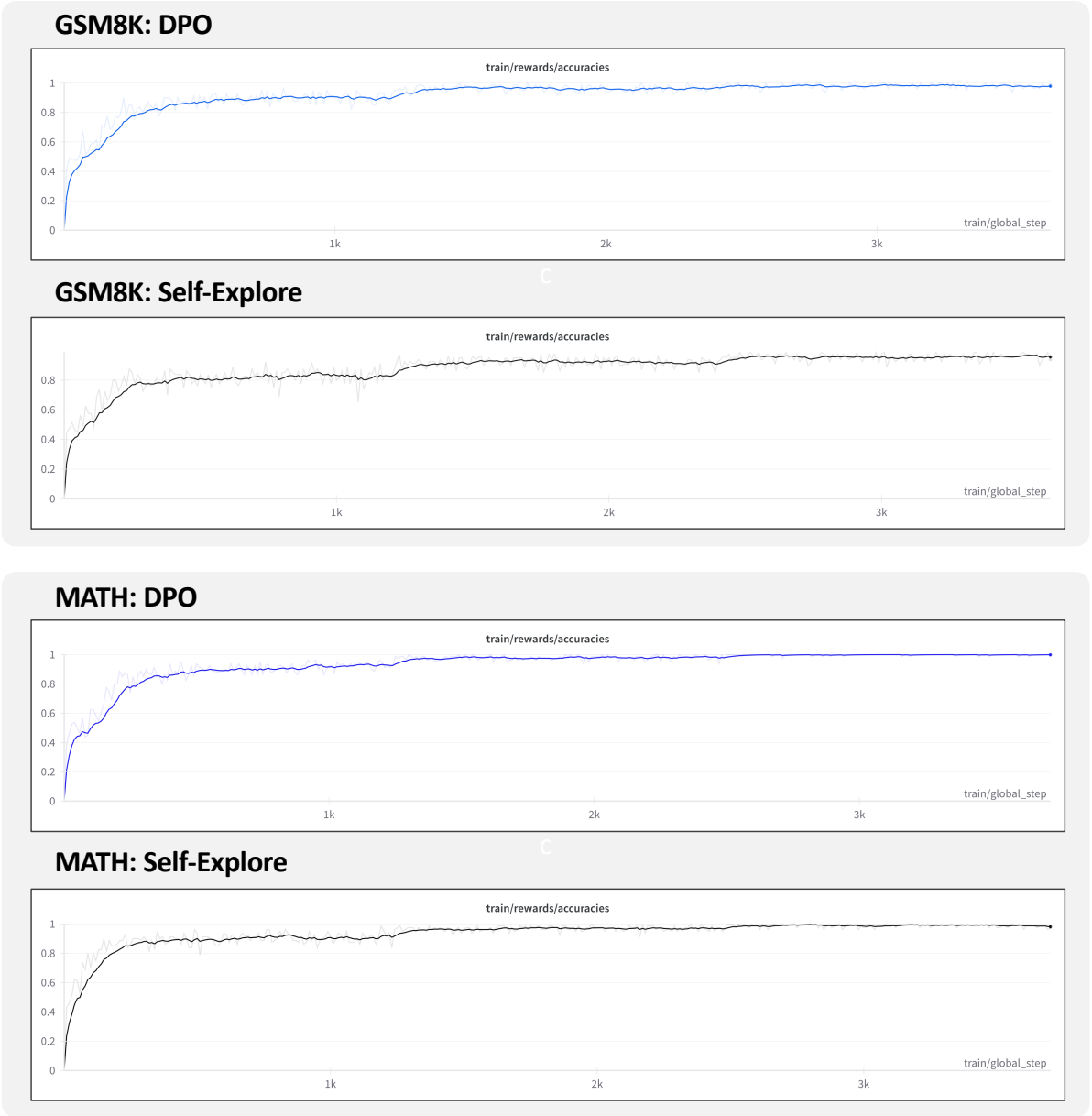**MATH: Self-Explore**

train/rewards/accuracies

Figure 8: Reward accuracy (*i.e.* winrate of *chosen* over *rejected* samples) of DPO and SELF-EXPLORE during training of DeepSeek-Math. For both methods, the accuracy quickly converges to 1 regardless of the supervision type.

Question

Randy has 9 oatmeal cookies, 4 chocolate chip cookies, and 5 sugar cookies. He ate 3 cookies for an early day snack, one of each flavor. He ate 2 oatmeal cookies for lunch. He gives 2 sugar cookies to his friends. Then, he bakes 4 of each flavor for dinner. How many cookies does he have now?

DPO

He has 600 movies and a third are in series so that's 600/3=<<600/3=200>>200 movies

He has 600 movies and 200 are in series so that means he has 600-200=<<600-200=400>>400 movies left

He has 400 movies and 40% are older movies so that's 400*.4=<<400*.4=160>>160 movies

He has 400 movies and 160 are older movies so that means he has 400-160=<<400-160=240>>240 movies left

He has 240 movies that are not in series or older and they cost $10 each so that's 240*$10=$<<240*10=2400>>2400

The answer is 2400

Self-Explore

He has 600 movies and a third are in series so that's 600/3=<<600/3=200>>200 movies

He has 600 movies and 200 are in series so that's 600-200=<<600-200=400>>400 movies

He has 400 movies and 40% are older movies so that's 400*.4=<<400*.4=160>>160 movies

He has 400 movies and 160 are older movies so that's 400-160=<<400-160=240>>240 movies

He has 240 movies that aren't in series or older and they cost $10 each so that's 240*10=$<<240*10=2400>>2400

He has 200 movies in series that cost $6 each so that's 200*6=$<<200*6=1200>>1200

He has 160 older movies that cost $5 each so that's 160*5=$<<160*5=800>>800

He has 2400+1200+800=$<<2400+1200+800=4400>>4400 The answer is 4400

Figure 9: Examples for sample generated by DPO and Self-Explore, respectively.

1464

We would like to request your feedback on the performance of the response of the assistant to the user instruction displayed below. In the feedback, I want you to rate the quality of the response in these 3 categories according to each score rubric:

[Skill 1. Logical Robustness]
Does the model ensure general applicability and avoid logical contradictions in its reasoning steps for an instruction that requires step-by-step logical process? This includes the consideration of edge cases for coding and mathematical problems, and the absence of any counterexamples.

Score 1: The logic of the model's response is completely incoherent.
Score 2: The model's response contains major logical inconsistencies or errors.
Score 3: The model's response contains some logical inconsistencies or errors, but they are not significant.
Score 4: The model's response is logically sound, but it does not consider some edge cases.
Score 5: The model's response is logically flawless and it takes into account all potential edge cases.

[Skill 2. Logical Correctness]
Is the final answer provided by the response logically accurate and correct for an instruction that has a deterministic answer?

Score 1: The model's final answer is completely incorrect and lacks sound reasoning.
Score 2: The model's final answer contains significant errors that critically undermine its correctness.
Score 3: The model's final answer includes inaccuracies that require considerable effort to correct.
Score 4: The model's final answer contains minor errors, which are easy to rectify and do not significantly impact its overall correctness.
Score 5: The model's final answer is completely accurate and sound.

[Skill 3. Logical Efficiency]
Is the response logically efficient? The logic behind the response should have no redundant step, remaining simple and efficient. For tasks involving coding, the proposed solution should also consider time complexity.

Score 1: The logic behind the response is significantly inefficient and redundant, necessitating a complete reorganization of logic for clarity and efficiency.
Score 2: The logic of the response lacks efficiency and conciseness, requiring a substantial reorganization for better optimization.
Score 3: The logic of the response is not efficient enough, necessitating major edits for improved optimization.
Score 4: The logic of the response is largely efficient, but it still has some redundant steps. It could be handled from minor edits for better optimization.
Score 5: The logic of the response is optimally efficient, requiring no further optimization.

[Instruction]
{question}

[Ground truth Answer]
{ground truth answer}

[Assistant's Response]
{answer}

[The End of Assistant's Response]

Please give feedback on the assistant's responses. Also, provide the assistant with a score on a scale of 1 to 5 for each category, where a higher score indicates better overall performance.

Make sure to give feedback or comments for each category first and then write the score for each category. Only write the feedback corresponding to the score rubric for each category. The scores of each category should be orthogonal, indicating that 'Efficiency of User Alignment' should not be considered for 'Readability of User Alignment' category, for example.

Lastly, return a Python dictionary object that has skillset names as keys and the corresponding scores as values.

Figure 10: GPT-4 prompt used for FLASK evaluation.

DPO

Logical Robustness: The assistant's response demonstrates a logical process in breaking down the number of movies into categories and calculating the cost for each category. However, it fails to include the calculations for the series movies and the older movies in the final summation, leading to an incomplete and therefore logically flawed conclusion. This oversight represents a significant logical inconsistency because it omits crucial steps in the problem-solving process, affecting the coherence of the solution. Therefore, the response's logic is not entirely coherent, as it neglects to account for all components of the problem statement. However, because the assistant correctly applies mathematical operations within each considered category, it does not fall to the lowest rating.
Score: 2

Logical Correctness: The assistant's final answer is incorrect because it only includes the cost for the 240 movies that are neither in series nor old. It neglects the cost for the 200 series movies and the 160 older movies, which were both explicitly mentioned in the instructions. Due to this significant oversight, the final answer provided is not just minorly inaccurate; it critically undermines the correctness of the response, as it only accounts for a portion of the total cost. Therefore, the provided answer significantly deviates from the accurate total.
Score: 2

Logical Efficiency: In the steps it does take, the assistant's logic is straightforward and direct, calculating the costs associated with each category of movies without unnecessary steps. However, since it fails to include all necessary calculations for a comprehensive answer, the efficiency of its logic is inherently compromised. The assistant's response is not burdened by redundant steps in the calculations it does perform, but the omission of critical steps leads to an incomplete solution. This oversight does not align with the highest standards of efficiency, as the most efficient response would include and correctly sum all relevant costs. Despite these issues, since the assistant does not introduce unnecessary complexity in the steps it takes, it does not warrant the lowest score in efficiency.
Score: 3

```python
{
    "Logical Robustness": 2,
    "Logical Correctness": 2,
    "Logical Efficiency": 3
}
```

Self-Explore

Logical Robustness: The assistant's response demonstrates a high degree of logical robustness. It accurately breaks down the problem into manageable steps, ensuring a clear and systematic approach to solving the problem. By calculating the costs for each category of movies (series, older, and normal) separately and then summing these to find the total cost, the response effectively addresses the problem without overlooking any details or introducing logical inconsistencies. It considers all potential categories of movies and their respective costs, thereby ensuring a comprehensive solution. Therefore, the logic employed in the assistant's response can be considered logically flawless in the context of the given instruction.
Score: 5

Logical Correctness: The assistant's final answer is completely accurate and logically sound. It correctly identifies the number of movies in each category (series, older, and neither), accurately calculates the cost for each category based on the given prices, and correctly sums these costs to arrive at the total cost of replacing the movies. The mathematical operations and the reasoning leading to the final answer are correct, making the response logically accurate in accordance with the requirements of the instruction.
Score: 5

Logical Efficiency: The response is logically efficient, following a straightforward and direct approach to solve the problem without introducing any unnecessary steps or complications. By sequentially addressing each movie category and calculating its cost before summing these costs, the assistant employs an efficient logical progression. This method not only simplifies the problem-solving process but also avoids any redundancy, making the logic behind the response optimally efficient for the task at hand.
Score: 5

```python
{
    "Logical Robustness": 5,
    "Logical Correctness": 5,
    "Logical Efficiency": 5
}
```

Figure 11: Results of GPT-4 FLASK evaluation for the generated solutions shown in Figure 9.