

# PROGRESS OR REGRESS? SELF-IMPROVEMENT REVERSAL IN POST-TRAINING

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Self-improvement through post-training methods such as iterative preference learning has been acclaimed for enhancing the problem-solving capabilities (e.g., mathematical reasoning) of Large Language Models (LLMs) without human intervention. However, as exploration deepens, it becomes crucial to assess whether these improvements genuinely signify progress in solving more challenging problems or if they could lead to unintended regressions. To address this, we propose a comprehensive evaluative framework that goes beyond the superficial pass@1 metric to scrutinize the underlying enhancements of post-training paradigms for self-improvement. Through rigorous experimentation and analysis across diverse problem-solving tasks, the empirical results point out the phenomenon of *self-improvement reversal*, where models showing improved performance across benchmarks will paradoxically exhibit declines in broader, essential capabilities, like output diversity and out-of-distribution (OOD) generalization. These findings indicate that current self-improvement practices through post-training are inadequate for equipping models to tackle more complex problems. Furthermore, they underscore the necessity of our critical evaluation metrics in discerning the *progress or regress* dichotomy for self-improving LLMs.

## 1 INTRODUCTION

In the rapidly evolving landscape of artificial intelligence (AI), the pursuit of self-improving large language models (LLMs) has garnered significant attention (Singh et al., 2023; Huang et al., 2023; Sun et al., 2024). The essence of self-improvement in LLMs lies in their capacity to iteratively refine models’ own performance without human intervention (Zelikman et al., 2022; Yuan et al., 2024). This capability is paramount as it holds the promise of fostering the development of more autonomous, adaptable, and efficient AI systems (Silver et al., 2016). Embracing and implementing self-improvement methodologies enables us to push the boundaries of these models’ capabilities, ultimately fostering the creation of more sophisticated and versatile AI applications (Significant-Gravitas, 2023).

Building on the concept of self-training (Grandvalet & Bengio, 2004), wherein models bootstrap their own generated responses for iterative training, a synergetic effect is observed and amplified. When models produce superior responses, the quality of the training data used to refine the models improves, subsequently enabling even better responses in future iterations. Such iterative post-training has become the standard paradigm for current self-improving AI (Yuan et al., 2024). Notably, STaR (Zelikman et al., 2022) has demonstrated that leveraging model’s self-generated reasoning steps for iterative supervised fine-tuning (SFT) can effectively enhance its reasoning abilities. Recent studies (Pang et al., 2024) have further revealed that employing iterative preference optimization in LLMs can achieve more performance improvements in reasoning tasks.

However, despite the promising advances in various post-training methods for self-improvement, a comprehensive understanding of their effects and underlying mechanisms is still lacking. To address this gap, in this study, we first endeavor to provide a comprehensive overview of the main iterative post-training paradigms for self-improvement, identifying the factors that contribute to consistent performance improvements. We decouple the influencing factors into the initial model, task datasets, the number of iterations, and the specific post-training methods employed. By isolating these variables, our comprehensive experiments and analysis uncover their individual and combined

054 effects on the model’s performance. This provides actionable insights for practitioners on how to  
055 perform iterative self-improvement practices more effectively.

056 While our extensive empirical results show that all these iterative post-training methods can achieve  
057 notable improvements in pass@1 accuracy across various problem-solving benchmarks, the evaluation  
058 has been limited to this single and superficial metric. Amidst the quest for self-improvement in  
059 LLMs, the persistent question arises: **are these iterative post-training methods truly fostering  
060 progress, or are they inadvertently leading to regression?** Transitioning beyond using pass@1  
061 accuracy as the indicator of improvement, we further develop an evaluative framework equipped  
062 with a comprehensive suite of metrics to assess improvement problems, solutions diversity, and  
063 OOD capabilities within the iterative process, enabling us to scrutinize the actual improvements  
064 beneath self-improvement. Surprisingly, our evaluation results display a paradoxical trend: as pass@1  
065 accuracy increases, the proposed metrics exhibit consistent performance declines.

066 The perceived reversals in our evaluative framework prompt a critical reflection on the effectiveness  
067 of current self-improvement practices. Through this study, we aim to illuminate the path forward for  
068 developing truly self-improving LLMs that balance accuracy, diversity, and robustness. To summarize,  
069 our work makes three significant contributions as follows:

070 • **Systematic Analysis:** In Sections 3 and 4, we systematically formulate current post-training  
071 methodologies and conduct extensive experiments to examine how various factors influence self-  
072 improvement in solving challenging tasks. To the best of our knowledge, this work provides the *first*  
073 in-depth overview of these influencing factors.

074 • **Metric Innovation:** In Section 5, we propose a comprehensive set of evaluation metrics to better  
075 capture the multifaceted nature of LLM performance in self-improvement practices.

076 • **Identified Phenomenon:** In Section 5, based on the proposed evaluation metrics, we reveal the  
077 phenomenon of *self-improvement reversal*, where increases in pass@1 accuracy compromise other  
078 essential capabilities such as solution diversity and OOD generalization.

## 081 2 BACKGROUND AND RELATED WORK

082 Training paradigms for LLMs typically consist of two stages: pre-training and post-training (Liu  
083 et al., 2024). Common post-training methods include supervised fine-tuning (Taori et al., 2023; Wang  
084 et al., 2023) and preference learning (Ouyang et al., 2022; Lee et al., 2024). Supervised fine-tuning  
085 trains LLMs to produce standard responses for given instructions, while preference learning trains  
086 LLMs to align with human preferences for different responses. Both methods, however, rely heavily  
087 on extensive human-annotated data.

088 An important question is whether effective LLM post-training can be achieved without excessive  
089 external feedback. Predating the era of LLMs, the self-training algorithm (Grandvalet & Bengio,  
090 2004; Goodfellow et al., 2014) demonstrated the potential to enhance model performance without  
091 additional labeled data. Recent studies have revived this concept, employing iterative self-training to  
092 facilitate self-improvement in LLMs without external feedback (Wang et al., 2023; Sun et al., 2023).  
093 For instance, STaR (Zelikman et al., 2022) shows that iterative training on the model’s own reasoning  
094 traces for correct answers can help solve increasingly difficult problems. Unlike the iterative nature  
095 of SFT, recent works (Yuan et al., 2024; Pang et al., 2024) propose iterative preference fine-tuning to  
096 aid models in self-improving.

097 In contrast to post-training methods, another line of research explores self-improvement through  
098 iterative post-prompting during inference (Huang et al., 2023). This approach does not update  
099 the model’s parameters but achieves self-improvement by generating reflections on its outputs and  
100 adjusting future outputs accordingly (Madaan et al., 2023; Gou et al., 2024). However, as revealed  
101 by Huang et al. (2024), post-prompting strategies are limited by the model’s *intrinsic self-correction*  
102 capabilities, thereby failing to significantly enhance problem-solving capabilities.

103 The potential of iterative post-training for self-improvement in LLMs remains underexplored. Al-  
104 though various post-training methods have demonstrated promise in general instruction-following  
105 tasks (Li et al., 2024; Sun et al., 2024; Chen et al., 2024; Yuan et al., 2024), they predominantly focus  
106 on aligning models with human values rather than enhancing the models’ internal knowledge. A key  
107 challenge remains whether LLMs can sustain consistent performance on more complex problem-

108 solving tasks. Recently, Pang et al. (2024) examined iterative preference learning in the context of  
 109 reasoning tasks, marking the first study to expand beyond instruction tuning.

110  
 111 Despite these advancements, a comprehensive overview investigating the effectiveness of various  
 112 iterative post-training methods for problem-solving is still lacking. **First**, it remains unclear how  
 113 improvements vary across iteration steps, different base models, task difficulties, and iterative post-  
 114 training techniques. For practitioners, there is a need for guidelines to help choose the most effective  
 115 post-training method among the various iterative post-training paradigms. **Second**, current research  
 116 only concentrates on maximizing benchmark scores through iterative self-improvement, there is little  
 117 exploration of the underlying factors contributing to performance gains. As a result, the progress and  
 118 reliability of different self-improvement methods are not guaranteed.

119 In this work, we aim to address these two critical issues. Our goal is not only to ensure the  
 120 effectiveness of various self-improvement methods but also to ensure that other capabilities are not  
 121 compromised during the complex self-improvement process.

### 123 3 POST-TRAINING FOR SELF-IMPROVEMENT

#### 125 3.1 FORMULATION

126  
 127 Consider a training dataset  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ , consisting of pairs of queries  $x_i$  and their correspond-  
 128 ing correct responses  $y_i$ . A foundation model, denoted as  $M_0$ . Our objective is to enhance  $M_0$   
 129 through a self-driven iterative post-training process, leveraging the model’s own outputs to refine its  
 130 capabilities, without reliance on external signals.

131  
 132 **Iterative Post-Training** The iterative post-training process involves a series of *post-training steps*,  
 133 each aimed at using the model’s previous outputs to guide its subsequent refinement. These steps are  
 134 designed to foster a continuous loop of self-improvement for the model.

135  
 136 The process is outlined across three main phases as follows, where the total number of iterations is  
 137 denoted as  $T$ , and the model employed in the  $t$ -th iteration is denoted as  $M_{t-1}$ , implying that  $M_0$  is  
 138 used in the first iteration:

139 • **Answer sampling:** In the  $t$ -th iteration, we prompt  $M_{t-1}$  to generate  $N$  answers for each query  
 140  $x_i$  in  $\mathcal{D}$  to form a new self-generated dataset  $\mathcal{D}_t^{\text{self}} = \{(x_i, y_i^j) | x_i \in \mathcal{D}, j = [1, N]\}$ .

141  
 142 • **Training set construction:** The training set  $D_t$  in the  $t$ -th iteration is assembled from  $\mathcal{D}_t^{\text{self}}$   
 143 without introducing any external data. The approach to constructing the training set depends on the  
 144 specific paradigm of post-training.

145 • **Model post-training:** Utilizing  $\mathcal{D}_t$ , the model  $M_{t-1}$  is refined into  $M_t$ .

146 It’s worth noting that, in the first iteration, we always directly supervised fine-tuning  $M_0$  on  $\mathcal{D}$  to  
 147 initialize  $M_1$  with task-specific knowledge.

148  
 149 Central to these diverse methodologies is the post-training function, symbolized as  $\mathcal{F}$ . We distinguish  
 150 among the practices based on the nature of  $\mathcal{F}$ , involving Supervised Fine-tuning (SFT) and Direct  
 151 Preference Optimization (DPO) (Rafailov et al., 2023), the latter being an effective implementation  
 152 of preference learning. During the SFT phase, this stage necessitates accurately labeled training data.  
 153 We derive these correct answers from  $\mathcal{D}_t^{\text{self}}$  to assemble the training dataset:

$$154 \mathcal{D}_t = \{(x_i, y^{\checkmark}) | \mathcal{R}(x_i, y^{\checkmark}) = 1, (x_i, y^{\checkmark}) \in \mathcal{D}_t^{\text{self}}\},$$

155  
 156 where  $\mathcal{R}(x, y)$  evaluates whether the answer  $y$  accurately addresses the question. In our problem-  
 157 solving task, the correctness of an answer  $y$  is verified by its alignment with the response provided in  
 158 the dataset. While during the DPO phase, for each query  $q_i$  in dataset  $\mathcal{D}$ , both correct and incorrect  
 159 responses from  $\mathcal{D}_t^{\text{self}}$  are paired to construct the training set, allowing for contrastive preference  
 160 learning:

$$161 \mathcal{D}_t = \{(x_i, y^{\checkmark}, y^{\times}) | \mathcal{R}(x_i, y^{\checkmark}) = 1, \mathcal{R}(x_i, y^{\times}) = 0, (x_i, y^{\checkmark}) \in \mathcal{D}_t^{\text{self}}, (x_i, y^{\times}) \in \mathcal{D}_t^{\text{self}}\}.$$

---

**Algorithm 1** Iterative Self-Improvement

---

**Require:** training set  $\mathcal{D} = \{x_i, y_i\}$ , base model  $M_0$ , iteration times  $T$ , post-training function series  $[\mathcal{F}_1(\cdot), \mathcal{F}_2(\cdot), \dots, \mathcal{F}_T(\cdot)]$

- 1:  $M_1 \leftarrow \text{SFT}((M_0)|\mathcal{D})$  ▷ Initialize base model with task-specific knowledge
- 2: **for**  $t = 2$  to  $T$  **do**
- 3:      $\mathcal{D}_t^{\text{self}} = \{(x_i, y_i^j) | x_i \in \mathcal{D}, y_i^j \sim M_{t-1}(x_i), j \in [1, N]\}$
- 4:     **if**  $\mathcal{F}(\cdot) == \text{SFT}$  **then**
- 5:          $\mathcal{D}_t = \{(x_i, y^{\checkmark}) | \mathcal{R}(x_i, y^{\checkmark}) = 1, (x_i, y^{\checkmark}) \in \mathcal{D}_t^{\text{self}}\}$
- 6:     **else**
- 7:          $\mathcal{D}_t = \{(x_i, y^{\checkmark}, y^{\times}) | \mathcal{R}(x_i, y^{\checkmark}) = 1, \mathcal{R}(x_i, y^{\times}) = 0, (x_i, y^{\checkmark}) \in \mathcal{D}_t^{\text{self}}, (x_i, y^{\times}) \in \mathcal{D}_t^{\text{self}}\}$
- 8:     **end if**
- 9:      $M_t \leftarrow \mathcal{F}_t(M_{t-1}|\mathcal{D}_t)$
- 10: **end for**

---

### 3.2 THREE ITERATIVE POST-TRAINING PARADIGMS

Through the implementation of designated self post-training steps (e.g., self-SFT), several distinct iterative post-training paradigms emerge. Our work focuses on three paradigms: (i) iterative SFT, where each cycle consists exclusively of self-SFT steps, (ii) Iterative DPO, characterized by successive self-DPO steps, except for the first iteration which supervised fine-tune the base model  $M_0$ , and (iii) iterative SFT-DPO, which initiates with a self-SFT step and alternates between self-DPO and self-SFT steps to form a complete iterative post-training loop.

We describe the unified procedure in Algorithm 1.

## 4 EXPERIMENT

As outlined in Algorithm 1, we hypothesize that the key variables—initialized model ( $M$ ), task dataset ( $\mathcal{D}$ ), iteration steps ( $T$ ), and post-training method ( $\mathcal{F}$ )—critically influence model performance during iterative self-improvement. This section explores the impact of these variables on different problem-solving tasks. We aim to determine if models consistently improve with increasing iterations ( $T$ ) and to uncover the trade-offs and comparative advantages of Iterative SFT, Iterative DPO, and Iterative SFT-DPO in enhancing performance across various tasks. Through this analysis, we seek to provide deeper insights into the mechanisms driving iterative self-improvement.

### 4.1 EXPERIMENTAL SETUP

**Datasets** To measure model problem-solving capabilities, we train and test on a broad spectrum of problem-solving datasets. We measure general knowledge using the CommonsenseQA (CSQA) (Talmor et al., 2019) dataset, assessing mathematical reasoning with the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) dataset, and weigh code generation skills using the MBPP dataset (Austin et al., 2021). Regarding the train-test split, we adhere to (Kojima et al., 2022), utilizing the validation set of CSQA for evaluation. The GSM8K and MATH datasets are employed with their predefined train-test splits. For the MBPP code dataset, we follow the approach outlined by (Austin et al., 2021) that utilizes examples of Task IDs 11-510 as the 500 test problems, and the remaining 374 examples ranging Task IDs from 601 to 974 for fine-tuning.

**Sampling and Rewarding** At the end of training, we sample  $N=50$  outputs for each problem using top  $p$  sampling (Holtzman et al., 2020) with  $p = 0.95$  and temperature 0.75. Considering the gold labels are provided for the problem-solving datasets, we use the correctness of final answer as a binary reward for the output to annotate the preference.

**Training** Our experiments primarily leverages three open-source models LLaMA-2-7B (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023) and LLaMA3-8B (AI@Meta, 2024), with a fully fine-tuning setting. For the implementation of preference-based learning, we utilize DPO (Rafailov et al., 2023) due to its scalability and efficiency. In each iteration, preference data are derived by sampling outputs from the newly updated model, utilizing an on-policy sampling strategy. Hence, we posit

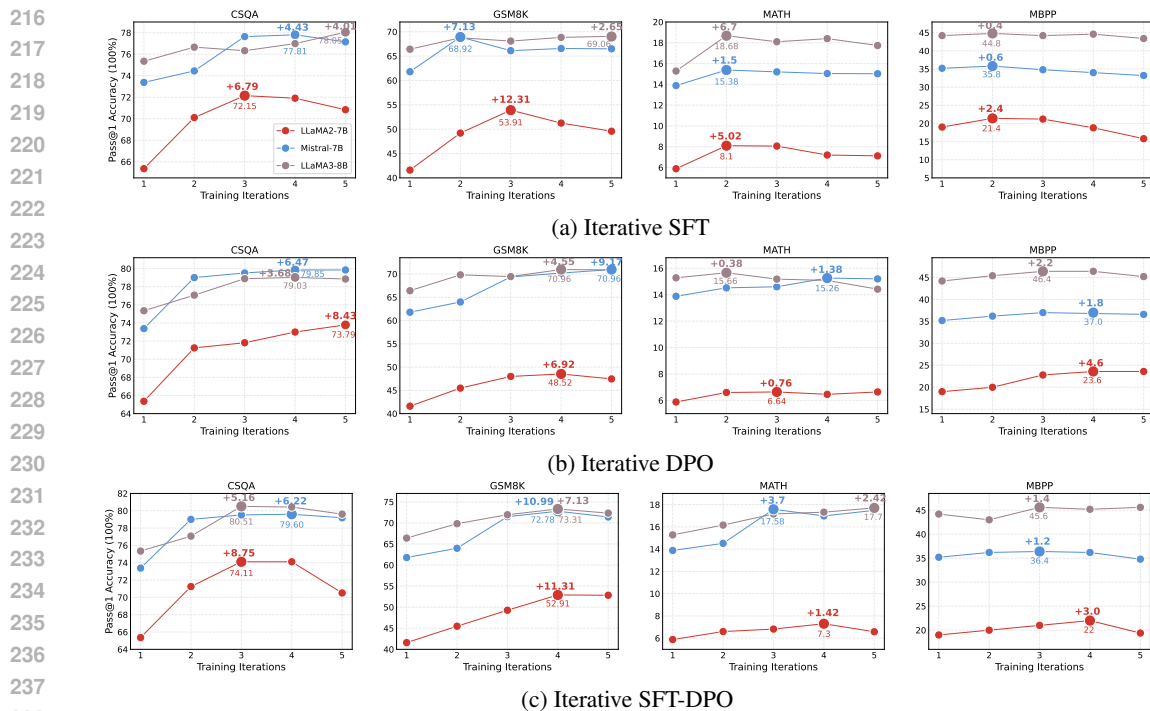


Figure 1: **Pass@1** accuracy across the four benchmarks by performing with the three paradigms: Iterative SFT, Iterative DPO and Iterative SFT-DPO. For each model along with the training iterations, we highlight the optimal result with a larger-size marker, the improvement above and final accuracy below.

that this online DPO can be treated as an effective and representative implementation for preference learning (Tajwar et al., 2024).

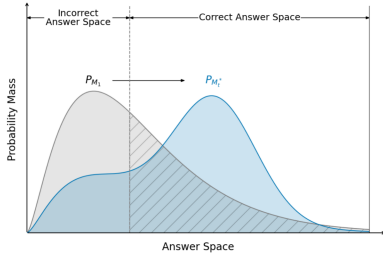
**Evaluation** We use greedy decoding as the temperature set 0 for testing generation. Meanwhile, we utilize zero-shot prompting (Kojima et al., 2022) for both answer sampling and evaluations since we find for LLMs finetuned on specific tasks, zero-shot prompting is superior to few-shot prompting. More experimental details can be seen in Appendix B.

## 4.2 MAIN RESULTS: DECOUPLING THE INFLUENCES OF VARIABLES

We perform the three post-training paradigms with the selected LLMs, training and testing them on the respective tasks. Based on the results shown in figure 1, we delve into the detailed analysis of how these variables influence the effectiveness of self-improvement.

**Iteration  $T$**  Across all methods and datasets, there is a general trend of improvement in pass@1 accuracy with increasing iteration steps. This indicates that iterative post-training effectively enhances model performance over time. However, the rate of improvement tends to plateau or even decline slightly after 4-5 iterations. This suggests that current post-training methods struggle to achieve long-lasting improvements, and excessive post-training (beyond a certain number of iterations) may even yield diminishing returns.

**Foundation Model  $M$**  The optimal accuracy improvements across various datasets and post-training methods suggest that LLaMA2-7B demonstrates a relatively higher capacity for improvement under iterative post-training. For instance, on the GSM8K dataset, LLaMA2-7B with Iterative SFT shows an improvement of +12.31 after 5 iterations, whereas LLaMA3-8B exhibits only a moderate gain. This indicates that the more capable  $M_1$  is not necessarily the model that achieves the most significant performance gains during the self-improvement process. However, the most capable model  $M_1$  generally achieves the highest optimal accuracy overall. For example, although LLaMA2-7B achieves



Model	Dataset	Coverage	$M_1$	$M_t^*$ Iterative SFT	$M_t^*$ Iterative DPO	$M_t^*$ Iterative SFT-DPO
LLaMA2-7B	CSQA	0.828	65.36	72.15	73.39	74.11
	GSM8K	0.398	41.6	53.91	48.52	52.91
	MATH	0.167	5.88	8.1	6.64	7.3
	MBPP	0.539	19.0	21.4	23.6	22.0
Mistral-7B	CSQA	0.875	73.38	77.81	79.85	79.6
	GSM8K	0.627	61.79	68.92	70.96	72.78
	MATH	0.143	11.56	15.38	15.26	17.58
	MBPP	0.629	35.2	35.8	37.0	36.4
LLaMA3-8B	CSQA	0.825	75.35	78.05	79.03	80.51
	GSM8K	0.802	66.41	69.06	70.96	73.31
	MATH	0.281	15.28	18.68	15.66	17.7
	MBPP	0.545	44.2	44.8	46.4	45.6

Figure 2: **Left:** The answer distributions of models.  $P_{M_1}$  and  $P_{M_t^*}$  represent the answer distributions of  $M_1$  and the optimal model  $M_t^*$  (achieving the highest pass@1 accuracy) within iterative process. The shaded area indicates the *correct answer coverage* of  $M_1$ . **Right:** For foundation model  $M$  and task  $\mathcal{D}$ , each line lists the correct answer coverage and the optimal pass@1 accuracy of  $M_t^*$  with the three iterative post-training methods. This table aims to display the relationship between correct answer coverage and the effectiveness of the post-training method  $\mathcal{F}$ .

the maximum gains on GSM8K with Iterative SFT, it still struggles to outperform LLaMA3-8B in terms of absolute optimal accuracy (53.91 vs. 69.06).

**Problem-solving Tasks  $\mathcal{D}$**  Models utilizing the three post-training methods all demonstrate notable improvements on the CSQA and GSM8K datasets, while showing more modest gains on the MATH and MBPP datasets. This indicates that, from the perspective of task difficulty, problems in the CSQA and GSM8K datasets are relatively easier for the models to resolve. In contrast, the MATH dataset poses significant challenges for 7B models due to its complexity. Additionally, the task of code generation, as represented by the MBPP dataset, is also difficult for these foundation models since they were not specifically pretrained on code domains.

**Comparative Analysis of Post-Training Methods** With foundation model  $M$  and task  $\mathcal{D}$  varying, the best-performing iterative method also changes accordingly. For example, for Mistral-7B on the CSQA dataset, Iterative-DPO achieves the highest accuracy improvement of +6.47. However, when applied to the GSM8K dataset, the Iterative SFT-DPO method yields the maximum improvement of +10.99. Therefore, with these identifiable variables characterized, it remains challenging for downstream practitioners to determine the optimal post-training method  $\mathcal{F}$  for their specific use case.

**Answer Coverage: Characterizing More Deciding Factor** As discussed above, the identifiable variables fail to provide clear clues on the effectiveness of the post-training method  $\mathcal{F}$  when foundation model  $M$  and task  $\mathcal{D}$  change. Upon closer examination of Figure 1, we find a common thread: regardless of the changes in  $M$  and  $\mathcal{D}$ , models ( $M_1$ ) that perform well on a task after the initial iteration of SFT tend to show substantial improvements with further iterations by performing iterative DPO and iterative SFT-DPO, compared to using Iterative SFT. Conversely, those  $M_1$  that achieve lower pass@1 accuracy initially exhibit limited gains with iterative DPO. Based on this observation, we hypothesize that  $M_1$ 's capability to solve the test problems fundamentally influences further improvement trends and optimal improvements of  $\mathcal{F}$ . To quantify  $M_1$ 's capability on the test set, we introduce **Correct Answer Coverage** as a measurement, the proportion of the correct answer space that the model's responses occupy. An illustrative display of this coverage is shown in Figure 2.

Mathematically, we can sample  $N$  model's outputs to approximate the answer space. As  $N \rightarrow \infty$ , these outputs can effectively represent the entire answer space. Therefore, expected accuracy over the  $N$  outputs can serve as an unbiased estimate of the *correct answer coverage*. Formally, we use the following equation to calculate  $M_1$  correct answer coverage (for a more detailed derivation, please refer to the Appendix C.):

$$\text{Correct Answer Coverage} = \mathbb{E} \left[ \frac{N_{\text{correct}}}{N} \right] \approx \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{1}{N} \sum_{i=1}^N \mathbb{I}[M(x_i) == y] \quad (1)$$

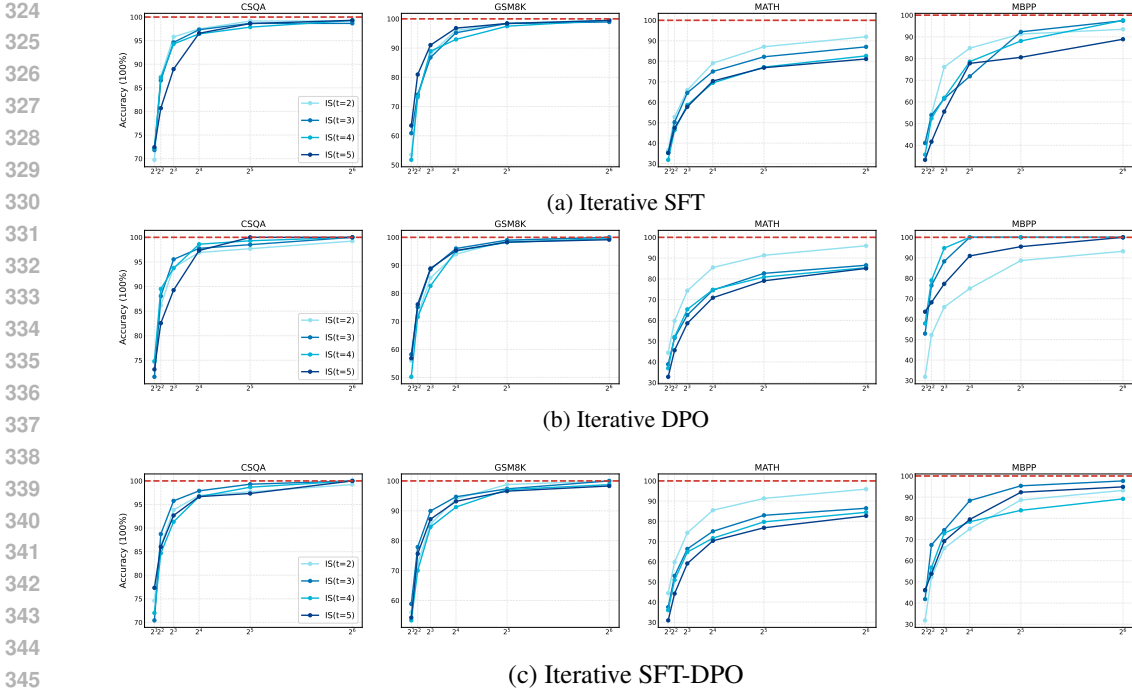


Figure 3: Pass@N accuracy of  $M_1$  with zero-shot prompting on  $IS(t)$ , for  $t > 2$ .

As shown in Figure 2, the relationship between correct answer coverage and optimal performance of  $\mathcal{F}$  validate our prior observation and hypothesis. The table clearly demonstrates that when the correct answer coverage is high ( $> 0.5$ ), Iterative DPO and Iterative SFT-DPO produce the best-performing  $M_t^*$ . Conversely, when the coverage is lower ( $\leq 0.5$ ), Iterative SFT is more effective in achieving the optimal  $M_t^*$ . Therefore, correct answer coverage can serve as a key factor in guiding practitioners to choose the most suitable iterative post-training method  $\mathcal{F}$  for the specific problem-solving task with a fixed foundation model.

## 5 CRITICAL EVALUATIONS ON SELF-IMPROVEMENT

Despite the extensive exploration of various post-training practices for self-improvement and a deepened understanding of their efficacy, current endeavors remain narrowly focused on enhancing performance numbers across these problem-solving benchmarks. Transitioning beyond using pass@1 accuracy as the indicator of improvement, our objective in this section is to engage in a critical examination and reevaluation of iterative self-improvement: discerning whether the improvements constitute genuine progress or merely regression. For brevity, all the results shown in this section is based on the foundation model  $M$  as Mistral-7B.

### 5.1 IMPROVEMENT PROBLEMS

In Figure 1, it is evident when  $t > 1$ , the pass@1 accuracy of  $M_t$  consistently improves in comparison to  $M_1$ . Traditionally, it has been assumed that this improvement indicates the model progressively learning to tackle more challenging problems (Zelikman et al., 2022). However, we posit a nuanced perspective: while an increase in pass@1 accuracy suggests improvements, it does not inherently equate to an increase in model capabilities to solve more difficult problems.

To better gauge how the model problem-solving capabilities evolve overtime, we propose to first quantify the *improvement problems* as **improvement set** (IS) at each iteration. An intuitive improvement between  $\mathcal{M}_t$  and  $\mathcal{M}_1$  is the pass@1 accuracy on test set, so we use the subset of test problems that  $\mathcal{M}_t$  correctly answers while  $\mathcal{M}_1$  fails under greedy decoding to represent  $IS(t)$ , defined as follows:

$$IS(t) = \{x \in \mathcal{D}_{\text{test}} \mid M_t(x) = y \wedge M_1(x) \neq y\}. \tag{2}$$



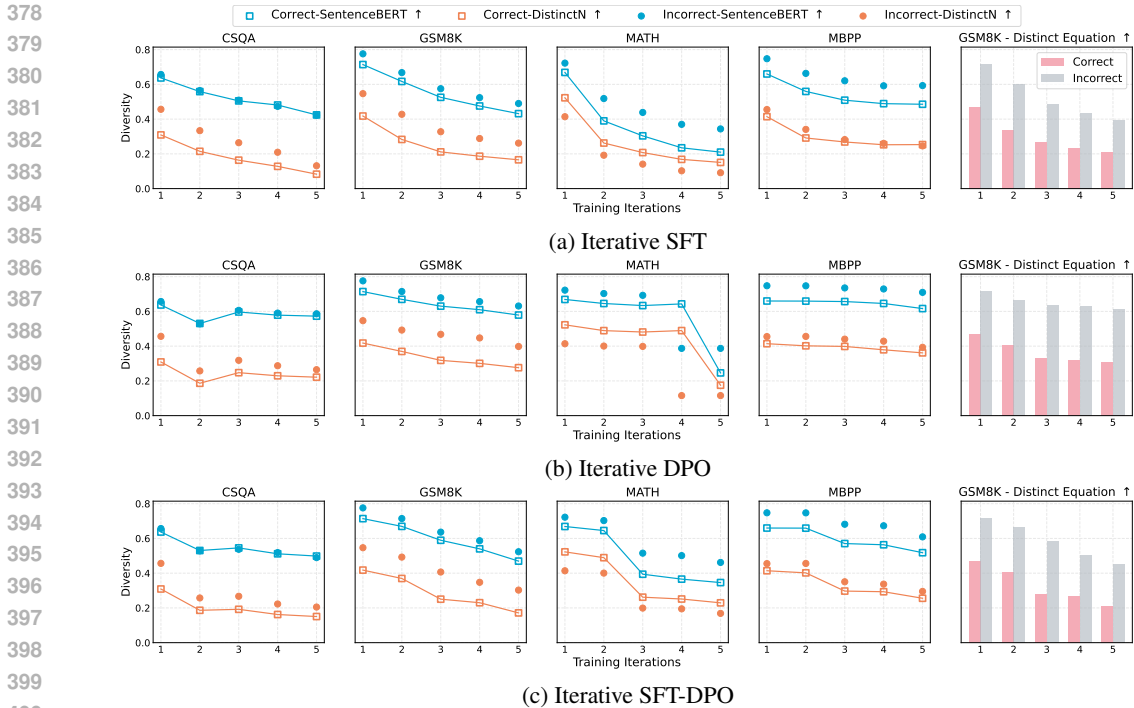


Figure 4: Diversity of the sampling outputs from  $M_t$  within the iterative process.

Then we can prompt  $M_1$  with the problems in IS(t) and sample  $N$  answers for each problem to record the pass@N accuracy of  $M_1$ . Notably, if  $M_1$  exhibits lower pass@N accuracy even as  $N$  increases, it can validate  $M_1$  struggles to solve the problems in IS(t) and the iterative process enhances the model’s problem-solving abilities.

We apply this evaluative methodology to CSQA, GSM8K, MATH and MBPP datasets with three post-training methods. Generation sampling  $N$  varies from  $2^1$  to  $2^6$  with the temperature set as 0.75.

**Reversal Observation** As depicted in Figure 3, contrary to prior assumptions, the rapid increase in pass@N accuracy with increasing  $N$  challenges the notion of progressively harder problem-solving. Specifically, as  $N$  grows,  $M_1$  achieves near-perfect pass@N accuracy on IS(t), suggesting its inherent capacity to tackle the deemed *improvement problems*.

**Selection Optimization for Answer Alignment** The empirical findings depicted in Figure 3 offer a critical insight: iterative self-improvement hardly entails the acquisition of new problem-solving abilities, but rather the enhancement of the model’s correct answer selection within its generation space.

## 5.2 SOLUTIONS DIVERSITY

While pass@1 accuracy measures the correctness of the final answer, it does not capture the diversity of solutions a model can generate. We posit that a model’s capacity to produce diverse solutions is indicative of its robustness and flexibility in problem-solving. To thoroughly understand the evolution of answer diversity during the process of iterative self-improvement, we employ a combination of **Distinct N-grams** (Li et al., 2016) and **Sentence-BERT embedding cosine similarity** (Reimers & Gurevych, 2019) to measure mod diversity. These metrics have been shown to correlate well with human assessments of diversity (Tevet & Berant, 2021). Additionally, for mathematical reasoning, we introduce **Distinct Equations** to measure the diversity of mathematical answers by analyzing the variety of equations in the generated solutions.

Each diversity metric  $Div$  takes a set of  $N$  model outputs, and produces a scalar score representing how diverse the set is. *Distinct N-grams* measures syntactic diversity by counting the number of unique n-grams (averaged over  $n = 1 \dots 5$ ) in the output set. The *Sentence-BERT* metric assesses



432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485

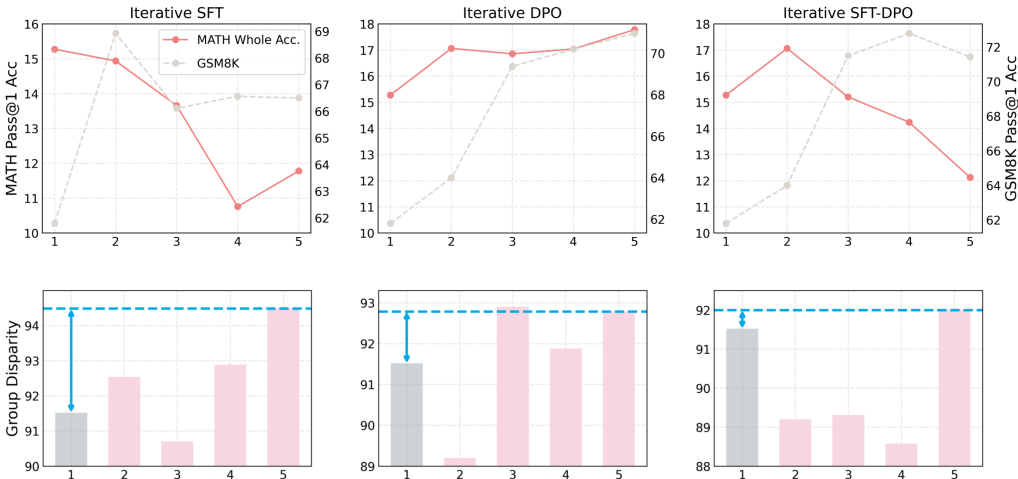


Figure 5: Pass@1 Accuracy of  $M_t$  on the MATH Algebra Test Set (Post-Training on GSM8K).

semantic diversity by embedding each output using a sentence transformer and calculating the average cosine similarity between embeddings. The metric is then 1 minus the average similarity, ensuring that higher scores reflect greater diversity. *Distinct Equations*, a specialized metric for mathematical reasoning, computes logical diversity by extracting all equations from the outputs and calculating the proportion of unique equations.

At each iteration, we sample  $N = 50$  outputs per problem with a temperature of 0.75. Outputs are categorized into correct and incorrect based on the final answer’s correctness. Then for each problem, we use the metric *Div* to calculate the average diversity for the correct and incorrect answers.

**Reversal Observation** Figure 4 presents the diversity results of three post-training methods during the iterative process. All methods show a consistent decrease in diversity, significantly diminishing the diversity of model outputs over iterations, impacting both correct and incorrect answers. This reduction is evident across all three metrics: syntactic, semantic, and logical diversity. Moreover, comparing Iterative SFT and Iterative DPO, it is clear that both methods exhibit a reduction in diversity, but the extent and pattern of reduction vary. For instance, Iterative DPO maintains a slightly higher semantic diversity (as measured by cosine similarity) over multiple iterations compared to Iterative SFT.

**Trade-Off with Output Diversity.** The evaluation results highlight a critical trade-off in iterative self-improvement: while aiming for higher accuracy, the diversity of outputs, which can be crucial for creativity and robustness in problem-solving, is compromised. Future approaches should consider strategies to maintain or even enhance diversity while improving accuracy.

### 5.3 OOD GENERALIZATION

In our pursuit to understand the broader implications of iterative self-improvement, it is crucial to assess not only the models’ performance on specific benchmarks but also their ability to generalize to out-of-distribution (OOD) tasks. Generalization performance provides insight into the robustness and adaptability of the models when faced with new and varying types of problems.

To evaluate the generalization capability of the models, we conducted iterative post-training on the GSM8K dataset and then transferred these models to the MATH algebra test set. The MATH algebra test set is organized into five levels of increasing difficulty, providing a comprehensive spectrum to analyze how well the models perform across groups with varied complexities.

For the sake of measuring OOD generalization, we define two metrics as follows defined to facilitate a deeper analysis:

- **Whole Accuracy (Whole Acc.):** This metric represents the pass@1 accuracy across the entire test set, encompassing all difficulty levels from Level 1 to Level 5.

486 • **Group Disparity:** This metric quantifies the difference in pass@1 accuracy between the best-  
 487 performing group (Level 1 test set) and the worst-performing group (Level 5 test set), thus highlighting  
 488 disparities in model performance across different difficulty levels. It is calculated using the following  
 489 equation:

$$490 \text{ Group Disparity} = \frac{\text{Pass}@1(\text{Level 1}) - \text{Pass}@1(\text{Level 5})}{\text{Pass}@1(\text{Level 1})} \quad (3)$$

492 A higher value of Group Disparity indicates that the model is performing significantly better on the  
 493 easier Level 1 while its performance deteriorates on the harder Level 5 group.

494 **Reversal Observation** As results shown in Figure 5, with the increase in iterative steps, Iterative  
 495 SFT and Iterative SFT-DPO can significantly harm the OOD generalization. In contrast, Iterative  
 496 DPO demonstrates a noticeable improvement, which may indicate better generalization to the OOD  
 497 test set, in consistent with the recent findings that DPO can improve OOD generalization (Kirk et al.,  
 498 2024). However, our more detailed examination of the results across Group Disparity shows Iterative  
 499 DPO is widening the performances between the easier and harder groups. This comparison uncovers  
 500 the OOD performance improvement from Iterative DPO actually stems from fitting simpler problems,  
 501 at the expense of solving more complex ones.

502 **Capabilities Collapse** All three iterative post-training methods can exacerbate the generalization  
 503 disparities across groups, inadvertently causing models to focus on easier problems rather than  
 504 enhancing their ability to solve more complex ones. As discussed in Section 5.2, the decrease in  
 505 solution diversity during iterations may be the bottleneck leading to reduced OOD generalization and  
 506 capability collapse. This highlights the intricate nature of model capabilities under self-improvement,  
 507 where capabilities at different levels and different facets will compromise each other. Therefore,  
 508 research developing more sophisticated methods should employ such a comprehensive, fine-grained  
 509 evaluative framework to monitor post-training processes, as an increase in a single facet of accuracy  
 510 does not necessarily represent true self-improvement.

## 511 6 EPILOGUE

512 **Conclusion** In this paper, we foster a comprehensive understanding of the current landscape of post-  
 513 training practices in self-improvement. Our evaluation, beyond simple pass@1 accuracy, utilizing  
 514 multifaceted metrics such as improvement problems, solutions diversity and OOD generalization,  
 515 underscores the necessity for a critical examination of both the progressive and regressive effects in  
 516 current self-improving post-training methods. By broadening the scope of our analysis, we provide  
 517 deeper insights into the true nature of iterative self-improvement with post-training, paving the way  
 518 for more robust and genuinely self-improving LLMs.  
 519

520 **Limitations and Future Work** Despite the comprehensive evaluation and nuanced insights pro-  
 521 vided by our study, there are several limitations to consider. Firstly, while our investigation covers a  
 522 variety of iterative post-training methods, the scope of our experiments is constrained by computa-  
 523 tional resources, limiting the range of models and tasks we could explore. Secondly, our evaluation  
 524 metrics, although more holistic than traditional measures, may still not capture all dimensions of  
 525 model performance and behavior, particularly in real-world applications. Thirdly, the iterative nature  
 526 of our methodologies requires extensive training cycles, which can be computationally expensive  
 527 and time-consuming, potentially limiting their practical applicability in environments with limited  
 528 resources.  
 529

530 Our future work would like to address the limitations identified in this study. Expanding the range of  
 531 models and tasks, particularly those involving more diverse and complex real-world scenarios, will  
 532 provide a more comprehensive understanding of iterative self-improvement. Additionally, developing  
 533 more sophisticated and multidimensional evaluation metrics will help in capturing the full spectrum  
 534 of model capabilities and limitations. Future studies could also explore optimizing the computational  
 535 efficiency of iterative post-training methods, making them more accessible for broader use. Moreover,  
 536 investigating the long-term impacts of these methodologies on model robustness and adaptability will  
 537 be crucial in ensuring sustainable advancements in LLM capabilities.  
 538

539

## REFERENCES

- 540  
541  
542 AI@Meta. Introducing meta llama 3: The most capable openly available llm to date. 2024. URL  
543 <https://ai.meta.com/blog/meta-llama-3/>.
- 544  
545 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan,  
546 Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language  
547 models. *arXiv preprint arXiv:2108.07732*, 2021.
- 548  
549 Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning  
550 converts weak language models to strong language models, 2024.
- 551  
552 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,  
553 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John  
554 Schulman. Training verifiers to solve math word problems, 2021.
- 555  
556 Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,  
557 Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
- 558  
559 Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen.  
560 CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth  
561 International Conference on Learning Representations*, 2024. URL [https://openreview.  
562 net/forum?id=Sx038qxjek](https://openreview.net/forum?id=Sx038qxjek).
- 563  
564 Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In L. Saul,  
565 Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 17.  
566 MIT Press, 2004. URL [https://proceedings.neurips.cc/paper\\_files/paper/  
567 2004/file/96f2b50b5d3613adf9c27049b2a888c7-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2004/file/96f2b50b5d3613adf9c27049b2a888c7-Paper.pdf).
- 568  
569 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,  
570 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*,  
571 2021.
- 572  
573 Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text  
574 degeneration. In *International Conference on Learning Representations*, 2020. URL [https:  
575 //openreview.net/forum?id=rygGQyrFvH](https://openreview.net/forum?id=rygGQyrFvH).
- 576  
577 Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,  
578 and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International  
579 Conference on Learning Representations*, 2022. URL [https://openreview.net/forum?  
580 id=nZeVKeeFYf9](https://openreview.net/forum?id=nZeVKeeFYf9).
- 581  
582 Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han.  
583 Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.),  
584 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp.  
585 1051–1068, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/  
586 v1/2023.emnlp-main.67. URL <https://aclanthology.org/2023.emnlp-main.67>.
- 587  
588 Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song,  
589 and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth  
590 International Conference on Learning Representations*, 2024. URL [https://openreview.  
591 net/forum?id=Ikmd3fKBPQ](https://openreview.net/forum?id=Ikmd3fKBPQ).
- 592  
593 Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot,  
594 Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al.  
595 Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- 596  
597 Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward  
598 Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation  
599 and diversity. In *The Twelfth International Conference on Learning Representations*, 2024. URL  
600 <https://openreview.net/forum?id=PXD3FAVHJT>.

- 594 Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large  
595 language models are zero-shot reasoners. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave,  
596 and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL  
597 <https://openreview.net/forum?id=e2TBb5y0yFf>.  
598
- 599 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.  
600 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model  
601 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating  
602 Systems Principles*, 2023.
- 603 Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret,  
604 Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. RLAIIF: Scaling reinforcement  
605 learning from human feedback with AI feedback, 2024. URL [https://openreview.net/  
606 forum?id=AAxIs3D2ZZ](https://openreview.net/forum?id=AAxIs3D2ZZ).  
607
- 608 Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting  
609 objective function for neural conversation models. In Kevin Knight, Ani Nenkova, and Owen  
610 Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the  
611 Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, San  
612 Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/  
613 N16-1014. URL <https://aclanthology.org/N16-1014>.
- 614 Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston,  
615 and Mike Lewis. Self-alignment with instruction backtranslation. In *The Twelfth International  
616 Conference on Learning Representations*, 2024. URL [https://openreview.net/forum?  
617 id=loiJHjBRsT](https://openreview.net/forum?id=loiJHjBRsT).  
618
- 619 Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi  
620 Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin  
621 Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. Understanding  
622 llms: A comprehensive overview from training to inference, 2024.
- 623 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon,  
624 Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,  
625 Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative  
626 refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing  
627 Systems*, 2023. URL <https://openreview.net/forum?id=S37hOerQLB>.  
628
- 629 Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong  
630 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton,  
631 Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and  
632 Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
- 633 Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason  
634 Weston. Iterative reasoning preference optimization, 2024.  
635
- 636 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea  
637 Finn. Direct preference optimization: Your language model is secretly a reward model. In  
638 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL [https://  
639 openreview.net/forum?id=HPuSIXJaa9](https://openreview.net/forum?id=HPuSIXJaa9).
- 640 Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-  
641 networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the  
642 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International  
643 Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong,  
644 China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410.  
645 URL <https://aclanthology.org/D19-1410>.  
646
- 647 Significant-Gravitas. Autogpt: Build & use ai agents. [https://github.com/  
SignificantGravitas/AutoGPT](https://github.com/SignificantGravitas/AutoGPT), 2023. Accessed: 2023-06-27.

- 648 David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche,  
649 Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman,  
650 Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P Lillicrap, Madeleine  
651 Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go  
652 with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/  
653 nature16961.
- 654 Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J.  
655 Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex  
656 Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor  
657 Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen  
658 Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi,  
659 Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan  
660 Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling  
661 self-training for problem-solving with language models, 2023.
- 662 Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming  
663 Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with  
664 minimal human supervision. In *Thirty-seventh Conference on Neural Information Processing  
665 Systems*, 2023. URL <https://openreview.net/forum?id=p40XRfBX96>.
- 666 Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Daniel Cox,  
667 Yiming Yang, and Chuang Gan. SALMON: Self-alignment with principle-following reward  
668 models. In *The Twelfth International Conference on Learning Representations*, 2024. URL  
669 <https://openreview.net/forum?id=xJbsmB8UMx>.
- 670 Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano  
671 Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal,  
672 on-policy data, 2024.
- 673 Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question  
674 answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and  
675 Tamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the  
676 Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and  
677 Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational  
678 Linguistics. doi: 10.18653/v1/N19-1421. URL <https://aclanthology.org/N19-1421>.
- 679 Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy  
680 Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model.  
681 [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca), 2023.
- 682 Guy Tevet and Jonathan Berant. Evaluating the evaluation of diversity in natural language generation.  
683 In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference  
684 of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp.  
685 326–346, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.  
686 eacl-main.25. URL <https://aclanthology.org/2021.eacl-main.25>.
- 687 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay  
688 Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation  
689 and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- 690 Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and  
691 Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In  
692 Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual  
693 Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–  
694 13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/  
695 2023.acl-long.754. URL <https://aclanthology.org/2023.acl-long.754>.
- 696 Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo  
697 Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for  
698 large language models. In *The Twelfth International Conference on Learning Representations*,  
699 2024. URL <https://openreview.net/forum?id=N8N0hgNDrt>.

702 Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason  
703 Weston. Self-rewarding language models, 2024.  
704

705 Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with  
706 reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.),  
707 *Advances in Neural Information Processing Systems*, 2022. URL [https://openreview.](https://openreview.net/forum?id=_3ELRdg2sgI)  
708 [net/forum?id=\\_3ELRdg2sgI](https://openreview.net/forum?id=_3ELRdg2sgI).

709 Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyang Luo, and Yongqiang  
710 Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint*  
711 *arXiv:2403.13372*, 2024. URL <http://arxiv.org/abs/2403.13372>.  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755

756 APPENDIX

757  
758 A ALGORITHMIC OVERVIEW OF LLM POST-TRAINING

759  
760 A.1 SUPERVISED FINE-TUNING

761 Supervised fine-tuning (SFT) is employed to tailor a pre-trained LLM to specific downstream tasks.  
762 Consider the training dataset  $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^N$ , where  $x^{(i)}$  is the problem and  $y^{(i)}$  is the target  
763 response, which the model  $M$  parameterized by  $\theta$  is trained to generate. The training objective of  
764 SFT is to minimize the following negative log-likelihood of the answers:  
765  
766

$$767 \mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}} \log p(y|x; \theta) \quad (4)$$

768 where  $p(y|x)$  is the probability of observing the answer  $y$  given the problem context  $x$ .

770  
771 A.2 PREFERENCE LEARNING

772 Preference learning is commonly used to train large language models to learn human preferences. The  
773 preference learning dataset includes not only problem and target response pairs but also preferences  
774 or rankings between different target responses for the given problem. A typical form of preference  
775 learning data is represented as  $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$ , where each piece of data contains a problem  
776  $x^{(i)}$ , and corresponding preferred and dispreferred responses, denoted  $y_w^{(i)}$  and  $y_l^{(i)}$ , respectively.  
777 Using a theoretical model of human discrete choice such as the Bradley-Terry model, which relates  
778 discrete choices to implicit goodness scores of the underlying options, we can train a reward model  
779 with maximum likelihood using this preference data. For the Bradley-Terry model, the reward  
780 modeling loss is:  
781  
782

$$783 \mathcal{L}_R(R_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} [\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))]. \quad (5)$$

784 In the context LLMs,  $r_\phi(x, y)$  is initialized from the SFT model  $\phi_{\text{SFT}}$ . Then, the learned reward  
785 function is used to provide feedback to the language model, through the optimization problem  
786 described below to train preferences in the language model:  
787

$$788 \max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} [R_\phi(x, y)] - \beta \mathbb{D}_{KL}[\pi_\theta(y|x) || \pi_{\text{SFT}}(y|x)], \quad (6)$$

789 where  $\beta$  is a parameter controlling the deviation from the base reference policy  $\pi_{\text{SFT}}$ . More recently,  
790 (Rafailov et al., 2023) show that the optimal policy for the learned reward can be extracted in  
791 closed form, especially skipping the need to perform iterative, approximate policy learning. The  
792 resulting algorithm, direct preference optimization (DPO), is simpler to tune and less computationally  
793 demanding than prior methods, while optimizing the same objective. We therefore use DPO as the  
794 algorithm for the implementation for preference learning in our experiments. The DPO loss for the  
795 language model policy  $\pi_\theta$  is  
796  
797

$$798 \mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_p} \left[ \log \left( \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{SFT}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{SFT}}(y_l | x)} \right) \right) \right]. \quad (7)$$

800  
801 B EXPERIMENTS

802  
803 B.1 TRAINING DETAILS

804 We use a fully fine-tuning setting for training LLaMA2-7B, Mistral-7B and LLaMA3-8B models  
805 either for supervised and preference fine-tuning. All training experiments are conducted on 8 NVIDIA  
806 A100 GPUs, and all experiments collectively consumed approximately 2000 A100 GPU hours. Our  
807 training codebase is based on LLaMA Factory (Zheng et al., 2024), and we use vLLM (Kwon  
808 et al., 2023) framework to perform inference for both CoT sampling and test evaluation. Detailed  
809 hyperparameters utilized throughout these experiments are documented in Table 1.



Type	Parameter	Value
Supervised Fine-Tuning	Batch Size	128
	Learning Rate {LLaMA2-7B}	$1e - 5$
	Learning Rate {Mistral-7B, LLaMA3-8B}	$2e - 6$
	Learning Rate Scheduler	Cosine
	Warm-up Ratio	0.03
	Optimizer	AdamW
	Epoch	3
Preference Fine-Tuning	Batch Size	32
	Learning Rate {LLaMA2-7B}	$2e - 6$
	Learning Rate {Mistral-7B, LLaMA3-8B}	$2e - 7$
	KL Coefficient ( $\beta$ )	0.3
	Optimizer	AdamW
	Epoch	1
Sampling Generation	Temperature	0.75
	Top_ $p$	0.95
	Top_ $k$	50
	Max_tokens	512
Evaluation Generation	Temperature	0
	Top_ $k$	-1
	Max_tokens	512

Table 1: Hyperparameters in all the experiments.

## B.2 DATASET DETAILS

**CommonsenseQA (CSQA)** (Talmor et al., 2019) offers a collection of 5-way multiple-choice questions on commonsense knowledge scenarios. It contains 12,102 questions with training/validation/testing set splits. Due to the unavailability of correct answers for the testing set, we utilize the validation set comprising 1,221 questions for evaluation, following the practice of (Kojima et al., 2022).

**GSM8K** (Cobbe et al., 2021) consists of 8.5K high-quality grade school math problems created by human problem writers, with the segmentation into 7.5K training problems and 1K test problems. These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations (+ - / \*) to reach the final answer.

**MATH** (Hendrycks et al., 2021) offers high school math competition problems that span seven subjects including Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra and Precalculus. It consists of 7,500 and 5,000 samples for training and testing, respectively. Compared to GSM8K, addressing MATH challenges involves more intricate and extended steps.

**MBPP** (Austin et al., 2021) consists of around 1,000 crowd-sourced Python programming problems, designed to be solvable by entry-level programmers, covering programming fundamentals, standard library functionality, and so on. Each problem consists of a task description, code solution and 3 automated test cases. Following the experimental setup described in (Austin et al., 2021), we utilize Task IDs 11-510, comprising 500 problems, as our test set. The remaining 374 problems, ranging from Task IDs 601 to 974, are employed for fine-tuning purposes.

## B.3 EVALUATION PROTOCOLS

**Zero-shot Prompting** We employ zero-shot prompts, as listed in Figure 6 for answer sampling and evaluation tests. Our comprehensive evaluation across all benchmarks demonstrates that zero-shot prompting not only reduces inference costs but also consistently outperforms few-shot prompting in terms of performance. Consequently, when LLMs are fine-tuned for task-specific applications, we advocate for the adoption of zero-shot prompting as a superior method compared to various few-shot

864 techniques. This perspective aligns with the findings of (Yu et al., 2024), who also reported the  
 865 advantages of zero-shot over few-shot prompting for fine-tuned LLMs.  
 866

```

867 CommonsenseQA
868 Question: {question}\nAnswer: Let's think step by step.\n
869
870 GSM8K
871 Question: {question}\nAnswer: Let's think step by step.\n
872
873 MATH
874 Question: {question}\nAnswer: Let's think step by step.\n
875
876 MBPP
877 You are an expert Python programmer, and here is your task: {question}
878 Your code should pass these tests:\n\n {test_list} \n[BEGIN]\n
  
```

881  
 882 Figure 6: Zero-shot Evaluation Prompt.  
 883

884 **Generation Diversity** In evaluating natural language generation (NLG) models, two prevalent  
 885 methods for assessing output diversity are the n-gram-based metric and the embedding-based metric,  
 886 which embeds generated sentences in a latent space. In this paper, we adopt distinct n-grams (Li  
 887 et al., 2016) and Sentence-BERT Embedding Cosine Similarity (Reimers & Gurevych, 2019) metrics.  
 888

889 **Distinct n-grams** is a straightforward yet effective method to quantify the lexical diversity of  
 890 generated text. This metric calculates the proportion of unique n-grams (sequences of n words)  
 891 within the generated text. The distinct n-gram measure is typically computed for unigrams, bigrams,  
 892 trigrams, and sometimes higher-order n-grams. Mathematically, for a generated sequence  $S$ , distinct-n  
 893 is defined as:

$$894 \text{distinct-n}(S) = \frac{|\text{unique n-grams in } S|}{|\text{total n-grams in } S|} \quad (8)$$

895 This measure provides a direct indication of how varied the vocabulary and phrases are within the  
 896 generated text. In general, higher distinct-n values indicate greater diversity.  
 897

898 **Sentence-BERT** embedding cosine similarity assesses the semantic diversity of generated sentences.  
 899 Sentences generated by the model are first encoded into embeddings using Sentence-BERT. The  
 900 cosine similarity between each pair of sentence embeddings is then computed. Cosine similarity  
 901 between two embeddings  $\mathbf{u}$  and  $\mathbf{v}$  is given by:

$$902 \cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (9)$$

903 The average cosine similarity of all sentence pairs gauges the overall semantic similarity. Lower  
 904 average cosine similarity indicates higher semantic diversity, as the sentences are less similar in  
 905 meaning. In our calculations, we measure diversity using  $1 - \text{average cosine similarity}$ , ensuring that  
 906 higher values reflect greater semantic diversity.  
 907

908 **Distinct Equations** provides a direct indication of how varied the mathematical approaches and  
 909 solutions are within the generated text. The calculation of this metric involves two steps that identifies  
 910 all equations present in the generated mathematical reasoning steps first, and then computes the ratio  
 911 of unique equations to the total number of equations. Mathematically, for a set of generated equations  
 912  $E$ , distinct equations is defined as:

$$913 D_{eq}(E) = \frac{|\text{unique equations in } E|}{|\text{total equations in } E|} \quad (10)$$

914 Higher  $D_{eq}$  values indicate greater diversity in the mathematical reasoning processes.  
 915  
 916  
 917

## C EXTRAPOLATION ANALYSIS ON COVERAGE

In Section 4.2, we mentioned that *correct answer coverage* may be a deeper factor influencing the subsequent improvements in iterative self-improvement. Here, we provide a detailed explanation of the related concepts involved in this influencing factor, as well as the derivation of the calculation for correct answer coverage (in Equation 1) as presented in this paper.

Additionally, we emphasize that our consideration of coverage as a deeper factor is a preliminary conclusion drawn from summarizing the factors of the model ( $M$ ), post-training function ( $\mathcal{F}$ ), and task dataset ( $\mathcal{D}$ ) and the empirical results validated in Figure 2. It should be noted that we need further work and more extensive experiments to both theoretically and empirically validate this observation, as the discussion of correct answer coverage is beyond our work. Our intention here is to offer empirical insights and lay the groundwork for future investigations into this aspect of iterative self-improvement.

**Answer space:** For a given dataset  $\mathcal{D}$  (test set), all possible (query, answer) pairs form the answer space of the test set. Here, the set of query is fixed, and for a given query, the number of possible answers can be quite large, hence we call the space as the *answer space*. Naturally, the entire answer space can be partitioned into a correct answer space and an incorrect answer space based on whether the answers are correct. In practical experiments, the correctness of an answer is approximated by whether its final result exactly matches the final result provided by the ground truth in the training set.

**Answer distribution:** For a given model  $M$ , its answer distribution refers to the probability distribution of generating answers conditioned on queries from dataset  $D$ . For a specific element  $(q_i, a_{ij})$ ,  $q_i \in \mathcal{D}_{\text{test}}$  in the answer space, the generation of this (query, answer) pair occurs in two steps: first, sampling  $q_i$  from all queries in  $D$ , then model  $M$  generates  $a_{ij}$  conditioned on  $q_i$ . Therefore, the probability at  $(q_i, a_{ij})$  is the product of the probability of sampling  $q_i$  from all queries in  $D$  and the probability of model  $M$  generating  $a_{ij}$  conditioned on  $q_i$ . Considering that all queries should have equal importance, we define that all queries are sampled with the same probability, which is  $\frac{1}{|\mathcal{D}_{\text{test}}|}$ . The answer distribution of  $M$  can be mathematically linked to model  $M$  as follows:

$$P_M(q_j, a_{ij}) \stackrel{\text{def}}{=} M(a_{ij}|q_j)P(q_j) = M(a_{ij}|q_j)\frac{1}{|\mathcal{D}_{\text{test}}|} \quad (11)$$

which  $P_M$  represents the model’s answer distribution, and  $M(a_{ij}|q_j)$  denotes the probability of model  $M$  generating  $a_{ij}$  conditioned on  $q_j$ .

**Correct Answer Coverage:** As mentioned earlier, the Correct Answer Coverage represents the correctness rate of all answers generated by model  $M$  on a dataset  $D$  (training set). It can be calculated using the following mathematical formula:

$$\text{Correct Answer Coverage} = \int_{\text{Correct Answer Space}} P_M(a, q) \quad (12)$$

Although we cannot exhaust the entire answer space and calculate a probability distribution to demonstrate the trend of the answer distribution in the progress of self-improvement, we can get an unbiased estimate of it by sampling answer and calculate the ratio of the number of correct answers to the total number of answers generated by model  $M$  for all queries in  $\mathcal{D}_{\text{test}}$ , where  $N$  answers are generated for each query, as illustrated in equation 1. The proof of Equation 1 is as follows:

972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025

$$\begin{aligned}
 \text{Correct Answer Coverage} &= \int_C P_M(a, q) \\
 &= \int_C \frac{1}{|\mathcal{D}_{\text{test}}|} P_M(a|q) \\
 &= \frac{1}{|\mathcal{D}_{\text{test}}|} \int_C P_M(a|q) \\
 &= \frac{1}{|\mathcal{D}_{\text{test}}|} \int_C \sum_{x \in \mathcal{D}_{\text{test}}} P_M(a|q = x) \\
 &= \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x \in \mathcal{D}_{\text{test}}} \int_C P_M(a|q = x) \\
 &= \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x \in \mathcal{D}_{\text{test}}} \mathbb{E} \left( \frac{N_x^c}{N} \right) \\
 &= \mathbb{E} \left( \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{N_x^c}{N} \right) \\
 &= \mathbb{E} \left[ \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x \in \mathcal{D}_{\text{test}}} \frac{1}{N} \sum_{i=1}^N \mathbb{I}[M(x_i) == y] \right]
 \end{aligned}
 \tag{13}$$

where  $C$  denotes the correct answer space, and  $N_x^c$  represents the number of correct answers generated by model conditioned on the given query  $x$ .

### D SCALING EXPERIMENTS

To validate that the phenomenon of self-improvement widely exists across different foundation models, ranging from 7B to more capable models, in this section, we scale the iterative self-improvement practices to LLaMA-2-70B (Touvron et al., 2023). As observed in Figure 2, Iterative SFT-DPO proves to be a robust practice that achieves consistent performance improvements regardless of the *correct answer coverage* being lower or higher. Considering the limitation of GPU resources, we hence set up the scaling experiment with the following parameters: the foundation model  $M$  is LLaMA-2-70B, the problem-solving task  $\mathcal{D}$  is GSM8K, the post-training function  $\mathcal{F}$  is Iterative SFT-DPO, and the number of iterations  $T = 5$ . Additionally, we employ quantized low-rank adaptation (LoRA) (Hu et al., 2022) for efficient post-training.

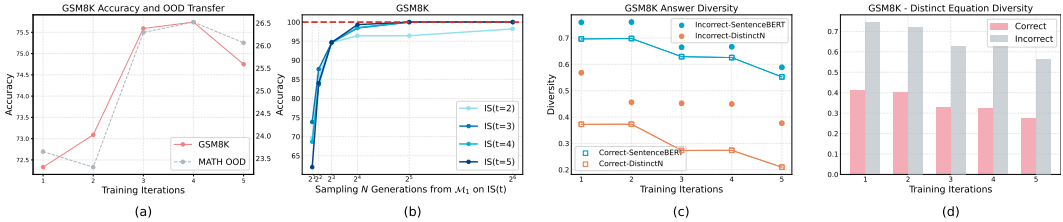


Figure 7: Perform iterative SFT-DPO on GSM8K (a), and then evaluate  $M_t$  with proposed metrics: pass@N on improvement problems (b), solutions diversity (c, d) and OOD generalization (a).

As shown in Figure 7, the performance of LLaMA2-70B on GSM8K demonstrates a similar self-improving trend, first descending to an optimal pass@1 accuracy and then declining after the fourth iteration. From Figure (b), it is evident that iterative self-improvement primarily involves the selection of correct answers within its generation space. Additionally, the solution diversity illustrated in Figures (c) and (d) highlights that the trade-off between pass@1 accuracy and output diversity is a universal phenomenon, even for highly capable 70B models.

Regarding OOD transfer accuracy shown in Figure (a), we observe that while performing Iterative SFT-DPO for self-improvement on GSM8K, LLaMA2-70B exhibits a certain degree of OOD gen-

1026 eralization to the more challenging MATH test set. This represents an *opposing trend* compared  
1027 with the whole accuracy of MATH shown in Figure 5. We conjecture that with the scaling of model  
1028 capacity, the capability of OOD generalization will gradually emerge and evolve within the iterative  
1029 self-improvement process.

1030

## 1031 E CASE STUDY

1032

1033 In this section, we select one problem from GSM8K test set and record the outputs of Mistral-7b  
1034 during the process of iterative DPO. This real case can vividly display our findings from Section 5,  
1035 which suggest that iterative self-improvement primarily manifests as an optimization of answer  
1036 selection rather than substantial enhancements in problem-solving capabilities. Additionally, we will  
1037 clearly see how diversity of the model’s reasoning steps decreases throughout the iterative process.

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133

<p><b>Question</b></p> <p>According to its nutritional info, a bag of chips has 250 calories per serving. If a 300g bag has 5 servings, how many grams can you eat if your daily calorie target is 2000 and you have already consumed 1800 calories?</p> <p><b>Annotated Answer</b></p> <p>If the total calorie target is 2000 and I have consumed 1800 calories then I have 2000-1800 = 200 calories left to eat. If each serving of chips has 250 calories and I only have 200 calories left to eat, then I can only eat 200/250 of a serving = 0.8 of a serving. We also know that a 300g bag of chips has 5 servings, hence each serving has 300g/5 = 60 grams. If I can only eat 0.8 of a serving, then I can eat only 60g * 0.8 = 48g = 48 grams ##### 48</p>	<p><b>Iteration 1</b></p> <p><b>Greedy Decoding Answer</b></p> <p>You have 2000-1800 = 200 calories left for the day. Since each serving has 250 calories, 5 servings have 250*5 = 1250 calories. Thus, you can eat 1250-200 = 1050 grams of chips. ##### 1050</p> <p><b>Sampling Answers</b></p> <p>A bag of chips has 5 servings, and you can eat 700/250 = 2.8 servings. Since a serving is 150g, you can eat 2.8*150 = 420g of chips. ##### 420</p> <p>If the bag has 5 servings, each serving is 250/5 = 50 calories. You have already consumed 1800 calories, so you can eat 2000-1800 = 200 calories more. Since each serving is 50 calories, you can eat 200/50 = 4 servings. Therefore, you can eat 4 servings * 300g/5 servings = 240g of chips ##### 240</p> <p>If a serving has 250 calories, then the entire bag has 250*5 = 1250 calories. Your daily calorie target is 2000, and you have already consumed 1800, so you still have 2000-1800 = 200 calories. Since there are 1250 calories in the bag of chips, then you can eat 200/1250 = 0.16 of the bag. There are 300g in the bag, so 0.16 of the bag is 300*0.16 = 48g ##### 48</p> <p>2000-1800 = 200 calories. 250 calories per serving x 5 servings = 250*5=1250 calories. 1250-200 = 1050 calories. So, you can eat 1050/250 = 4 servings. Thus, you can eat 4 servings x 300g/serving = 1200g of chips. ##### 1200</p> <p>If a serving of chips has 250 calories, then 1000 calories contain 1000/250 = 4 servings. If you have already consumed 1800 calories, 2000-1800=200 calories are left. Since 1 serving has 250 calories, 200/250 = 0.8 servings can be eaten. Since 1 serving is equal to 50 grams, then 0.8 servings x 50 grams/serving = 40 grams can be eaten. ##### 40</p>	<p><b>Iteration 2</b></p> <p><b>Greedy Decoding Answer</b></p> <p>A bag of chips has 250 calories per serving and there are 5 servings in a bag, so a bag has 250*5 = 1250 calories. You have already consumed 1800 calories and your daily calorie target is 2000, so you have 2000-1800 = 200 calories left. A bag of chips has 1250 calories and you have 200 calories left, so you can eat 1250-200 = 1050 grams of chips ##### 1050</p> <p><b>Sampling Answers</b></p> <p>You still have 2000 - 1800 = 200 calories left for the day. Each serving has 250 calories so 200/250 = 0.8 servings. 0.8 servings is 0.8 x 300 = 0.8*300=240 grams. ##### 240</p> <p>If a 300g bag has 5 servings, each serving is 300/5 = 60g. If a bag of chips has 250 calories per serving, then 60g of chips have 250*60/100 = 150 calories. If you have already consumed 1800 calories, then you can eat 2000-1800 = 200 calories worth of chips. Since 60g of chips have 150 calories, then 200 calories worth of chips are 200/150*60 = 120g ##### 120</p> <p>Since a bag of chips has 5 servings and each serving has 250 calories, the whole bag has 5*250 = 1250 calories. If you have already consumed 1800 calories and your daily calorie target is 2000, then you can still eat 2000-1800 = 200 calories. Since a bag of chips has 1250 calories, then you can eat 200/1250 = 0.16 of a bag. Since a bag of chips weighs 300g and you can only eat 0.16 of a bag, then you can eat 0.16*300 = 48g ##### 48</p> <p>A serving has 250 calories so 5 servings have 5*250 = 1250 calories. You have already consumed 1800 calories so you have 2000-1800 = 200 calories left to consume. You can eat 200/250 = 0.8 servings. One serving is 150g so you can eat 0.8*150 = 120 grams ##### 120</p>
<p><b>Iteration 3</b></p> <p><b>Greedy Decoding Answer</b></p> <p>If a serving has 250 calories, 5 servings will have 250*5 = 1250 calories. If you have already consumed 1800 calories, you can still eat 2000-1800 = 200 calories. Since 1 serving has 250 calories, you can eat 200/250 = 0.8 servings. Since each serving is 50g, you can eat 0.8*50 = &lt;0.8*50=40&gt;40g ##### 40</p> <p><b>Sampling Answers</b></p> <p>If a serving has 250 calories, then a bag of chips has 250*5=1250 calories. If you have already consumed 1800 calories, then you have 2000-1800=200 calories left for the day. Therefore, you can eat 200/250=0.8 of a bag of chips. In grams, this would be 0.8*300=240 ##### 240</p> <p>If a 300g bag has 5 servings, each serving is 300/5 = 60g. The nutritional info says a serving has 250 calories, so 60g has 250*60/60 = 250 calories. If you have already consumed 1800 calories, you have 2000-1800 = 200 calories left. Since 60g has 250 calories, you can eat 200/250 = 0.8*60 = 48g of chips. ##### 48</p> <p>If a serving has 250 calories, then the whole bag has 250*5 = 1250 calories. If you have already consumed 1800 calories, then you can eat 2000-1800 = 200 calories worth of chips. Since there are 1250 calories in the whole bag, then 200 calories is equal to 200/1250 = 1/6 of the bag. Therefore, you can eat 300/6 = 50 grams of chips. ##### 50</p>	<p><b>Iteration 4</b></p> <p><b>Greedy Decoding Answer</b></p> <p>If a bag of chips has 250 calories per serving, and a 300g bag has 5 servings, then the 300g bag has 250*5 = 1250 calories. If you have already consumed 1800 calories, then you can eat 2000-1800 = 200 calories worth of chips. Since each chip has 1250 calories, you can eat 200/1250 = 0.16 bags of chips. Since each bag of chips has 300g, you can eat 0.16*300 = 48 grams of chips. ##### 48</p> <p><b>Sampling Answers</b></p> <p>If a serving has 250 calories, then a bag of chips has 250*5=1250 calories. If you have already consumed 1800 calories, then you have 2000-1800=200 calories left for the day. Therefore, you can eat 200/250=0.8 of a bag of chips. In grams, this would be 0.8*300=240 ##### 240</p> <p>If a 300g bag has 5 servings, each serving is 300/5 = 60g. If a bag of chips has 250 calories per serving, then 60g of chips have 250*60/60 = 250 calories. If you have already consumed 1800 calories, then you can eat 2000-1800 = 200 calories worth of chips. Since 60g of chips have 250 calories, then 200/250 = 0.8 servings of chips can be eaten. Therefore, you can eat 0.8*60 = 48g of chips. ##### 48</p> <p>A serving has 250 calories and there are 5 servings in the bag, so the whole bag has 250*5 = 1250 calories. You have already consumed 1800 calories and your daily target is 2000, so you can still consume 2000-1800 = 200 calories. The bag has 1250 calories and you can still consume 200 calories, so you can eat 1250-200 = 1050 grams. ##### 1050</p>	<p><b>Iteration 5</b></p> <p><b>Greedy Decoding Answer</b></p> <p>If a serving has 250 calories, and a bag has 5 servings, then the bag has 250*5 = 1250 calories. If you've already consumed 1800 calories, then you can eat 2000-1800 = 200 calories worth of chips. Since each chip has 1250 calories, then you can eat 200/1250 = 0.16 bags of chips. Since a bag has 300g, then you can eat 0.16*300 = 48g of chips. ##### 48</p> <p><b>Sampling Answers</b></p> <p>If a 300g bag has 5 servings, each serving is 300/5 = 60g. If your daily calorie target is 2000 and you have already consumed 1800 calories, you can consume 2000-1800 = 200 calories more. Since each serving has 250 calories, you can consume 200/250 = 0.8 servings more. Since each serving is 60g, you can consume 0.8*60 = 48g more. ##### 48</p> <p>If a 300g bag has 5 servings, each serving is 300/5 = 60g. If your daily calorie target is 2000 and you have already consumed 1800 calories, you have 2000-1800 = 200 calories left. Since a bag of chips has 250 calories per serving, you can eat 200/250 = 0.8 servings. Since each serving is 60g, you can eat 0.8*60 = 48g of chips. ##### 48</p> <p>A bag of chips has 250 calories per serving, and there are 5 servings in a 300g bag, so the 300g bag has 250*5 = 1250 calories. If a 300g bag has 1250 calories, and you have already consumed 1800 calories, then you can eat 1250-1800 = -550 calories worth of chips. Since there are 1000 calories in 1 kilogram, then -550 calories is equal to 1000*-0.55 = 550 grams. ##### 550</p>

Figure 8: One case of sampled responses from the test set after iterative DPO training of the Mistral-7B model on the GSM8K dataset.