REFINING ANSWER DISTRIBUTIONS FOR IMPROVED LARGE LANGUAGE MODEL REASONING

Soumyasundar Pal[*] **Didier Chételat**[*] **Yingxue Zhang**[*] **Mark Coates**[†]

ABSTRACT

Large Language Models (LLMs) have exhibited an impressive capability to perform reasoning tasks, especially if they are encouraged to generate a sequence of intermediate steps. Reasoning performance can be improved by suitably combining multiple LLM responses, generated either in parallel in a single query, or via sequential interactions with LLMs throughout the reasoning process. Existing strategies for combination, such as self-consistency and progressive-hint-prompting, make inefficient usage of the LLM responses. We present Refined Answer Distributions, a novel and principled algorithmic framework to enhance the reasoning capabilities of LLMs. Our approach can be viewed as an iterative sampling strategy for forming a Monte Carlo approximation of an underlying distribution of answers, with the goal of identifying the mode – the most likely answer. Empirical evaluation on several reasoning benchmarks demonstrates the superiority of the proposed approach.

1 INTRODUCTION

As Large Language Models (LLMs) have increased in size, they have demonstrated increasing reasoning abilities (Brown et al., 2020), despite not being explicitly trained to reason (Wei et al., 2022a). In particular, Chain-of-Thought (CoT) prompting has become standard for eliciting these abilities, either through few-shot examples (Wei et al., 2022b) or via a triggering sentence such as "Let's think step by step." (Kojima et al., 2022). Nevertheless, although LLMs often produce correct reasoning steps, they struggle with higher-level planning (Saparov and He, 2022), motivating researchers to explore strategies to remedy this deficiency. An effective solution is to sample several chains-of-thoughts and take the most common answer as the final vote, an approach called Self-Consistency (CoT+SC) (Wang et al., 2023). However, despite its impressive empirical performance, the gains quickly plateau on many benchmarks, often with no improvement after five samples (Aggarwal et al., 2023). Thus, a more complex reasoning strategy appears necessary.

One promising direction involves encouraging LLMs to iteratively refine their reasoning, like humans often do (Shinn et al., 2023; Li et al., 2023; Gou et al., 2023; Madaan et al., 2023). However, Huang et al. (2024) demonstrate that the capability and effectiveness of LLMs' self-correction is overstated in the existing literature due to the use of oracle labels for determining stopping criteria (Shinn et al., 2023), unfair experimental protocols (Du et al., 2023), and sub-optimal initial prompt design (Madaan et al., 2023). Moreover, the review/feedback prompts employed in these approaches are often long and complex, and include intricate, hand-crafted examples, tailored for specific domains or benchmarks. In spite of such extensive prompt engineering, Huang et al. (2024) observe that most of these approaches perform worse than self-consistency in a fair evaluation setting.

In this paper, we propose a novel iterative strategy called *Refined Answer Distributions (RAD)* that offers a more principled and practical way of reasoning with refinement. We consider a setting where we can conduct LLM calls sequentially or in parallel. Our method does not make major changes to the initial prompt in later calls, and we do not need extensive prompting effort to invoke an LLM's review of the previous answers. The process starts by constructing an initial distribution of answers using CoT. In subsequent rounds, we incorporate the unique answers from the previous round in the prompt. This leads to a new collection of answers, which we use to refine the answer distribution via a marginalization process. Our approach is agnostic to the strategy for incorporating a previous answer in the prompt, provided that it satisfies a 'probability flow' condition that we specify. In our numerical experiments, we show that an existing hint-based prompting strategy (Zheng et al., 2023)

[*]Huawei Noah's Ark Lab, Canada, [†]McGill University, Mila, & ILLS

satisfies this condition for a broad spectrum of reasoning datasets. By maintaining a distribution, we reduce sampling variance and make more efficient usage of the LLM calls.

We make the following contributions:

- We introduce a novel iterative refinement strategy for reasoning with LLMs, with the key differentiator that the method maintains and updates a *distribution* over answers. Our work highlights that an LLM can indeed derive benefit from self-reflecting on distributions of its past answers when attempting reasoning tasks, without a need to resort to extensive prompt design or hand-crafted examples.
- Via multiple experiments with GPT-3.5 Turbo (Brown et al., 2020), GPT-4 Turbo (OpenAI et al., 2024), the cost efficient GPT-4o-mini, and Llama models (Grattafiori et al., 2024), we show that our proposed approach leads to consistently improved reasoning performance compared to *state-of-the-art* baselines for the same number of LLM calls and comparable token cost. We conduct experiments carefully to ensure there is no evaluation bias in favour of methods that employ refinement. Notably, out of 36 experimental scenarios, we observe that the proposed RAD variants have the highest accuracy in 30.
- We show in the experiments that our approach is flexible in that it can be combined successfully with different strategies for obtaining an initial distribution of answers (e.g. Chain of Thoughts Wei et al. (2022b), Progressive Hint Prompting (PHP) Zheng et al. (2023)).

2 Problem Statement

Let x be a question or a task in natural language, described in one or more sentences. Its true answer is denoted y, which can take different forms depending on the context, such as a number, a True/False boolean variable, or an option (a)/(b)/(c) from a multiple-choice set. Potentially, we also assume to have access to a (small) set of triplets $\mathcal{I}{=}\{(x_j, z_j, y_j)\}_{j=1}^{K}$ corresponding to semantically-similar questions x_j, answers y_j, and rationales z_j. Each rationale z_j is a sequence of short sentences that describe the step-by-step reasoning process leading to the answer y_j.

We assume that we can query the LLM in series or in parallel. Our task is to design a strategy for prompting the LLM and combining the responses to provide an answer $\hat{y}$ for the question x. Performance is measured in terms of the average accuracy of the response, i.e., $\mathbb{E}[\mathbf{1}(\hat{y} = y)]$ for the indicator function $\mathbf{1}$.

3 Methodology

When presented with the question, an LLM produces a random answer $\tilde{y}$, drawn from an internal distribution that is dependent on the prompt and the LLM's parameters. To avoid notational clutter, we suppress these dependencies and denote this distribution by $p(\tilde{y}|x)$. This distribution is analytically intractable but one can sample from it directly by prompting the LLM and subsequently collecting its answer.

The reasoning ability of the LLM, i.e., the probability of producing the correct answer, is improved by careful construction of the prompt. For example, an encouragement to produce an explanation/rationale in the form of a sequence of short sentences to describe the step-by-step reasoning process has been shown to ameliorate LLMs' performance significantly compared to direct prompting (Wei et al., 2022b). We denote the provided rationale as z, so the response of the LLM is a pair $(z, \tilde{y})$. If rationale-annotated in-context examples are available, then reasoning can be improved by incorporating in the prompt a (small) set in the form of triplets $\mathcal{I}{=}\{(x_j, z_j, y_j)\}_{j=1}^{K}$.

Viewing the LLM response as a sample from the distribution, we can hypothesize that, if the LLM is capable of effective reasoning for the presented question, the mode of the distribution is most likely to be the correct answer. We would therefore like to extract the mode. One approach is to sample, either in parallel or sequentially, multiple LLM responses (each containing a rationale and answer). We can then select the answer corresponding to the Monte Carlo estimate of the mode by taking a majority vote over the sampled responses (Wang et al., 2023).

It has been observed that LLM output can be improved via a refinement or self-reflection process (Zheng et al., 2023; Wu et al., 2024; Li et al., 2023; Madaan et al., 2023; Park et al., 2023). In this process, the LLM is provided with its previous response and/or answer, and asked to take it into account, or criticize it, before producing a refined response.

This observation is the cornerstone of our proposed methodology. Rather than seeking the mode of the original distribution $p(\tilde{y}|x)$, we construct a sequence of distributions $\{p_r(\tilde{y}|x)\}_{r>1}$, where each successive distribution in the sequence is constructed via a refinement process using samples from the previous distribution, as the number of interactions with the LLM, r, grows. This refinement process involves marginalization over the LLM's previous answers, which are refined in the current iteration. We initialize $p_1(\tilde{y}|x) = p(\tilde{y}|x)$, i.e., we start with the distribution of answers obtained from the first interaction with the LLM at $r = 1$. Our hypothesis is that the probability of the correct answer, $p_r(y|x)$, increases with r, so the mode of a distribution later in the sequence, i.e., $r > 1$, is more likely to be correct than the mode of $p(\tilde{y}|x)$.

3.1 Intuition

We now provide an example to illustrate why marginalizing over previous answers should make the mode of the inference distribution more likely to be the correct answer. Suppose that we are presented with a binary question x with answer, say, $y = 1$, and let us say that the probability of the correct answer is initially relatively low, $p(\tilde{y}{=}1|x) = 0.4$. However, when we provide the correct answer on the prompt and ask the LLM to refine it, the LLM is much more likely to answer correctly, $p(\tilde{y}{=}1|x, \text{Refine}(y{=}1)) = 0.8$. Refining the incorrect answer 0 also strongly tilts the LLM towards that answer, but crucially, with slightly less probability, $p(\tilde{y}{=}0|x, \text{Refine}(y{=}0)) = 0.6$. This is not unexpected or unusual, because it is often easier to see the truth of a statement in hindsight (or verify rather than solve unaided). With our proposed marginalization procedure, the updated distribution of the answer would be:

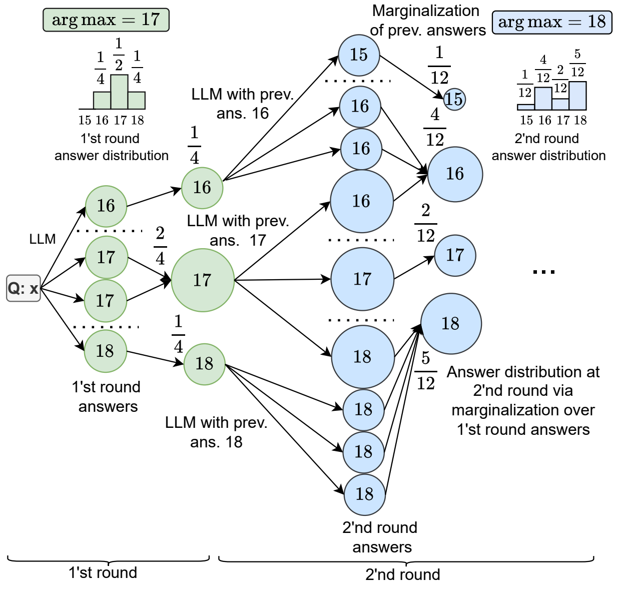


Figure 1: Illustration of one iteration of our proposed method, Refined Answer Distribution (RAD). At its initialization, a distribution of answers is obtained from the LLM via multiple queries. In each subsequent iteration, new answers are sampled by refining each distinct old answer. The resulting samples are then accordingly weighted by the probability of the previous answers for marginalization.

$$\begin{aligned} p_2(\tilde{y}{=}1|x) &= p_1(\tilde{y}{=}1|x)p(\tilde{y}{=}1|x, \text{Refine}(y{=}1)) + p_1(\tilde{y}{=}0|x)p(\tilde{y}{=}1|x, \text{Refine}(y{=}0))\,, \\ &= 0.4 \times 0.8 + (1 - 0.4) \times (1 - 0.6) = 0.56 > 0.4\,. \end{aligned}$$

Not only is the probability higher than before, but crucially, the mode of the distribution now aligns with the right answer ($y = 1$).

More generally, this augmentation will be observed **if and only if** the flow of probability mass into the correct answer exceeds the flow of probability mass out of the correct answer. The flow out is $p_1(\tilde{y}{=}y|x)\big(1 - p(\tilde{y}{=}y|x, \text{Refine}(y))\big)$, whereas the flow in is $\sum_{y' \neq y} p_1(\tilde{y}{=}y'|x)p(\tilde{y}{=}y|x, \text{Refine}(y'))$. Since we expect $p_1(\tilde{y}{=}y|x, \text{Refine}(y))$ to be close to 1, the flow out is likely to be small. By contrast, we might anticipate that when the LLM is presented with an incorrect answer, it can often ignore it to a large extent. Let us assume that $p(\tilde{y}{=}y|x, \text{Refine}(y')) > cp_1(\tilde{y}{=}y|x)$ for all y' for some positive constant $c < 1$. Then the flow in exceeds $cp_1(\tilde{y}{=}y|x)(1 - p_1(\tilde{y}{=}y|x))$. Thus, if $p(\tilde{y}{=}y|x, \text{Refine}(y)) > 1 - c(1 - p_1(\tilde{y}{=}y|x))$, the mass assigned to the correct answer will increase. For example, consider $p_1(\tilde{y}{=}y|x)) = 0.4$ and $c = 0.3$. Then we need $p(\tilde{y}{=}y|x, \text{Refine}(y)) > 1{-}0.3{\times}(1{-}0.4) = 0.82$.

We also note that repeated application of this procedure is further advantageous, which motivates the iterative version of our algorithm. We formalize this intuition into a general procedure and provide an algorithm for approximating these distributions next.

3.2 Refined Answer Distributions

Our approach updates the distribution of answers by marginalizing over the answers obtained at the previous iteration. We denote the conditional probability of yielding $\tilde{y}$ as the answer for task x with a previous answer y' by $p(\tilde{y}|x, \mathrm{Refine}(y'))$. We define a sequence of distributions $\{p_r(\tilde{y}|x)\}_{r>1}$, where two successive distributions are related as follows:

$$p_{r+1}(\tilde{y}|x) = \int p(\tilde{y}|x, \mathrm{Refine}(y'))p_r(y'|x)\,dy'\,. \tag{1}$$

The integral is replaced by a sum when $\tilde{y}$ is discrete, e.g., for multiple-choice questions.

Implementation: We now outline the steps for performing one iteration of RAD. As a concrete example, Figure 1 illustrates the procedure of approximating $p_2(\tilde{y}|x)$ from $p_1(\tilde{y}|x)$ in detail. Since neither $p(\tilde{y}|x, \mathrm{Refine}(y'))$ nor $p_r(\tilde{y}|x)$ can be computed analytically, we need to resort to a Monte Carlo approach for estimating $p_{r+1}(\tilde{y}|x)$.

Suppose, at the end of the r-th iteration, $p_r(\tilde{y}|x)$ is approximated as follows:

$$p_r(\tilde{y}|x) \approx \sum_{m=1}^{M} \omega^m \delta(\tilde{y} - y^m)\,, \tag{2}$$

where $\delta(\cdot)$ is the Kronecker delta function, $\{y^m\}_{m=1}^{M}$ is the set of distinct answers, and ω^m is the estimated probability of obtaining the answer y^m under the distribution $p_r(\cdot|x)$. For example, in Figure 1, at $r = 1$, we have $M = 3$ distinct answers $y^1 = 16, y^2 = 17$, and $y^3 = 18$ with estimated probabilities $\omega^1 = \frac{1}{4}, \omega^2 = \frac{1}{2}$, and $\omega^3 = \frac{1}{4}$. If the correct answer $y = 18$, then the LLM's current answer $\hat{y} = 17$, based on the estimated mode of $p_1(\tilde{y}|x)$, is incorrect.

Assuming a sampling budget of B_{r+1}, which denotes the maximally allowed number of answers to be sampled at the $(r+1)$-th iteration, we modify, for each $m = 1, \ldots, M$, the prompt by appending y^m as the previous answer, and sample $\lfloor \frac{B_{r+1}}{M} \rfloor$ answers subsequently. This forms the following Monte Carlo approximation:

$$p(\tilde{y}|x, \mathrm{Refine}(y = y^m)) \approx \sum_{\ell=1}^{L_m} \bar{\omega}^{\ell,m} \delta(\tilde{y} - y^{\ell,m})\,. \tag{3}$$

Here, $\{y^{\ell,m}\}_{\ell=1}^{L_m}$ are the L_m distinct answers extracted from the $\lfloor \frac{B_{r+1}}{M} \rfloor$ answers and $\bar{\omega}^{\ell,m}$ is the estimated probability of having $y^{\ell,m}$ as the answer conditioned on the previous answer y^m. In Figure 1, the total budget for $r = 2$, i.e. $B_2 = 9$, the number of distinct answers for different previous answers are $L_1 = 2, L_2 = 3$, and $L_3 = 1$. For the previous answer $y^1 = 16$, the estimated conditional probabilities of the answers $y^{1,1} = 15$ and $y^{2,1} = 16$ are $\bar{\omega}^{1,1} = \frac{1}{3}$ and $\bar{\omega}^{2,1} = \frac{2}{3}$ respectively.

Using equations 2 and 3, we can approximate equation 1 as follows:

$$p_{r+1}(\tilde{y}|x) \approx \sum_{m=1}^{M}\sum_{\ell=1}^{L_m} \omega^m \bar{\omega}^{\ell,m} \delta(\tilde{y} - y^{\ell,m}) = \sum_{n=1}^{N} \bar{\omega}^n \delta(\tilde{y} - \bar{y}^n)\,. \tag{4}$$

Here, N is the number of distinct answers among $\{y^{\ell,m}\}_{\ell=1,m=1}^{L_m,M}$. The probability of having $\bar{y}^n$ as the answer is estimated as:

$$\bar{\omega}^n = \sum_{m=1}^{M}\sum_{\ell=1}^{L_m} \omega^m \bar{\omega}^{\ell,m} \mathbf{1}(y^{\ell,m} = \bar{y}^n)\,. \tag{5}$$

From Figure 1, we observe that at $r = 2$, we have $N = 4$, the distinct answers are $\bar{y}^1 = 15, \bar{y}^2 = 16, \bar{y}^3 = 17$, and $\bar{y}^4 = 18$. As shown in eq. 5, the probability of obtaining $\bar{y}^4 = 18$ is $\bar{\omega}^4 = (\frac{2}{4} \times \frac{1}{3}) + (\frac{1}{4} \times 1) = \frac{5}{12}$. We observe that the probability of obtaining the correct answer is increased in one round of RAD.

We can stop this procedure by applying a variety of stopping criteria. For example, we can stop (i) after a fixed number of iterations (when $r>R$); or (ii) based on a predefined sampling budget B_{max} (when $r>R$, for R such that $\sum_{p=1}^{R} B_p \leqslant B_{max} < \sum_{p=1}^{R+1} B_p$); or (iii) when the estimate of the mode of $p_r(y|x)$ remains the same for two successive iterations. Algorithm 1 in Appendix A provides a pseudocode description.

Discussion: Intuitively, refining some previous answers is more likely to lead to the correct answer than others for a given reasoning task. Our framework naturally defines the usefulness of an answer y' at the end of the r-th iteration by the probability $p_r(y'|x)$, and subsequently weights the answers generated by refinement of y' at the $(r+1)$-th iteration by this value, while updating the distribution of answers in eq. 1.

We also note that the RAD framework is agnostic to the choice of prompts and is generally applicable to any advanced prompting technique, as those methods combined with SC can be used to initialize $p_1(\tilde{y}|x)$ for subsequent RAD iterations. Our contribution is thus orthogonal to prompt engineering approaches. In our implementation of RAD in Section 4, we adopt the hint-based prompting strategy of (Zheng et al., 2023) for the refinement of previous answers. Beyond the measurement of reasoning accuracy, we conduct a detailed empirical analysis across multiple benchmarks and LLMs (Figures 3, 4, and 5, Tables 6 and 7), which provides statistically significant evidence that the use of this prompt satisfies the 'probability flow' criterion specified in Section 3.1.

4 Experimental Results

Benchmarks: We evaluate the proposed RAD algorithm on six arithmetic benchmarks: AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2015), SingleEQ (Koncel-Kedziorski et al., 2015), SVAMP (Patel et al., 2021), GSM8K (Cobbe et al., 2021), and AQuA (Ling et al., 2017). AddSub and SingleEq contain easier problems, whereas the tasks in MultiArith, SVAMP, GSM8K, and AQuA are more challenging. In addition, we conduct experiments on the MATH (Hendrycks et al., 2021) dataset, which consists of a large collection of significantly more difficult mathematical questions of seven subcategories. In order to demonstrate the general applicability of RAD beyond mathematical reasoning, we also consider two BIG-Bench Hard Suzgun et al. (2023) tasks, namely Date Understanding and Object Tracking. More details of the datasets are deferred to Appendix B.

Models: We use five different language models: GPT-3.5 Turbo (Brown et al., 2020), which was fine-tuned using RLHF from a previous version (GPT-3), its upgraded version GPT-4 Turbo (OpenAI et al., 2024), the more recent cost-efficient GPT-4o-mini, and two Llama-based models, Llama-3-8b-instruct and Llama-3-70b-instruct (Grattafiori et al., 2024). All three GPT models are closed-source, but can be publicly accessed using the OpenAI API[1]. The Llama models are open-source, although in practice, we used a commercial API service[2].

Baselines and Experimental Setting: We compare our approach to few-shot CoT (Wei et al., 2022b), its combination with SC (Wang et al., 2023), PHP (Zheng et al., 2023), and PHP+SC. We refer to the proposed algorithm as CoT+RAD, since the same few-shot prompt as CoT is employed to initialize our approach. For relatively cheaper LLMs, GPT-3.5 Turbo and GPT-4o-mini, we also consider another variant of our method called PHP+RAD, where the initial answer distribution is obtained from several PHP provided answers (i.e., PHP+SC). We also include known results of alternative iterative refinement methods on the same models and datasets, namely Self-Refine (Madaan et al., 2023), CRITIC (Gou et al., 2023), repeated introspection (Self-Convinced prompting (Zhang et al., 2023a)), Multi-Agent Debate (Du et al., 2023), multi-agent multi-model round table conference (ReConcile (Chih-Yao Chen et al., 2023)), and verification methods such as Self-Verification (Weng et al., 2023) and Forward-Backward reasoning (FOBAR (Jiang et al., 2024)). Computational budget limitations prevented us from running every possible combination of model, benchmark and competitor, especially since none of these works include results on GPT-4-Turbo and GPT-4o-mini. However, Huang et al. (2024) thoroughly investigated many of these methods and found these approaches systematically inferior to a simple Self-Consistency baseline (CoT+SC), which is corroborated by our experimental results. For the MATH dataset, we compare our approach with a recently proposed multi-agent prompting technique MACM (Lei et al., 2024), which progressively performs each intermediate computational step (akin to a thought in CoT), verifies its correctness using code, and determines whether it can help in reaching the final answer via several agent interactions. In order to avoid prohibitive token cost, we only use GPT-4o-mini for the MATH dataset. Additionally, we restrict the use of Llama models to the arithmetic benchmarks. We conduct our experiments on an Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz.

[1]`https://openai.com/api`

[2]`https://replicate.com/meta/meta-llama-3-70b-instruct`

For a fair comparison with CoT+SC, which requires sampling of multiple CoTs, we ensure that the proposed RAD uses a comparable number of CoTs. We use a total budget of B_{max}=40 sampled CoTs in two iterations of CoT+RAD, with B_1=5, B_2=15, and B_3=20. We allocate more CoTs to the later iterations ($r > 1$), since we need to estimate $p(\tilde{y}|x, \text{Refine}(y'))$ for multiple values of y'. As we initialize $p_1(\tilde{y}|x)$ with CoT+SC, increasing the number of CoTs does not contribute substantially to improved performance at $r = 1$ (Aggarwal et al., 2023). For PHP+RAD, we perform one iteration of marginalization with B_1=20 and B_2=20.

For the CoT+SC algorithm, we sample exactly 40 CoTs to report performance, as in Wang et al. (2023). For PHP, generating one answer requires at least 2 interactions, but the exact number of CoTs cannot be known beforehand. Therefore, in order to ensure a fair comparison, we collect PHP answers in the PHP+SC algorithm until the total number of LLM calls matches that of CoT+RAD, which ensures that PHP+SC has an inference time comparable to that of CoT+RAD. Except for CoT and PHP, which use greedy decoding, a temperature of 0.7 is used for all sampling based approaches, following the experimental settings of Wang et al. (2023) and Zheng et al. (2023). The answer extraction and cleansing is carried out by following the same steps laid out by Kojima et al. (2022). Additionally, for all datasets except AQuA (where the answers are multiple choice between A-E), we use a 3'rd decimal rounding off of LLM answers and 'ground truth' before comparing them. This fixes some questions in most of those five arithmetic datasets and the MATH dataset for all competing algorithms, (e.g. the 'true' answer is 0.066666, but the LLM's answer is 0.067), where the LLM's answer is essentially correct, but is declared incorrect due to a rounding error. A symbolic evaluation using latex2sympy2[3] is carried out to determine the correctness of the final answer for the MATH dataset (e.g. $2x$+7 is equivalent to 7+$2x$). We measure the accuracy of the answer as the performance metric. CoT employs the same 4-shot prompt for AQuA and the same 8-shot prompt for other four arithmetic datasets, as designed by Wei et al. (2022b). For the MATH dataset and the BBH tasks, we use the same prompts as Zheng et al. (2023) and Suzgun et al. (2023) respectively. PHP and PHP+SC also use the same base prompts to obtain the initial answer(s). Example prompts for all algorithms can be found in Appendix G.

Results on Arithmetic Benchmarks: We summarize the experimental results using the GPT models in Table 1. Results using the weaker Llama models can be found as Table 4 in Appendix C. For each dataset and LLM, we conduct a Wilcoxon signed rank test between the top two algorithms and declare their difference statistically significant at the 5% level. As we use more recent versions of the GPT models than in the original articles of CoT+SC (Wang et al., 2023) and PHP (Zheng et al., 2023), the results are not directly comparable, but are broadly in line with their reported numbers. We observe that for all LLMs, with or without SC, PHP achieves higher accuracy than CoT prompting in most cases, demonstrating the advantage of using the LLMs' answers as hints. The superior accuracy of CoT+SC compared to the greedy decoding of CoT for the majority of datasets showcases the strong empirical performance of SC, arising due to the consideration of diverse reasoning paths. PHP+SC emerges as a close competitor to CoT+SC in most cases, although the relative accuracy gain compared to PHP is much lower, since PHP in itself is a strong baseline. Since PHP+SC does not consistently outperform CoT+SC, we can conclude that the incorporation of hints alone is insufficient to achieve better reasoning accuracy.

Our approach, CoT+RAD, considerably outperforms CoT+SC in most cases. The PHP+RAD variant performs comparably to CoT+RAD on GPT-3.5 Turbo but shows improved performance on GPT-4o-mini. This shows that our RAD approach is generally applicable, as it can be combined with different prompting methods for initialization, and it is not overly sensitive to the choice of hyperparameters.

One benchmark that deviates from this pattern is AQuA using GPT-4 Turbo, where the best performing procedure is CoT+SC. This might be due to the fact that AQuA is the only multiple-choice question-answering benchmark among the six, and the employed hinting prompt "The answer is close to A)" makes less sense for these types of questions. Further research on how to better extend PHP's hinting prompt to these types of problems might be valuable. In addition, all methods perform only as well as (or even worse than) a vanilla few-shot CoT and PHP on AddSub for both GPT-3.5 Turbo and GPT-4 Turbo models, possibly indicating the fact that the gains to be had using advanced methods on a dataset containing relatively simple questions are rather limited.

[3]`https://pypi.org/project/latex2sympy2/`

Table 1: Mean and standard error of accuracy (in %) of few-shot arithmetic reasoning. The **highest** accuracy among all competing algorithms using the same LLM is marked in **bold** and is shown in **red**, **blue**, and **orange** for **GPT-3.5 Turbo**, **GPT-4 Turbo**, and **GPT-4o-mini** respectively. The second-best accuracy in those cases is marked with an underline and is shown in light red, light blue, and light orange respectively. The **highest** accuracy is marked with an asterisk if the difference from the second-best accuracy is statistically significant.

LLM	Algorithm	AddSub	MultiArith	SingleEQ	SVAMP	GSM8K	AQuA
GPT-3.5 Turbo	CoT	91.4$\pm$1.4	97.8$\pm$0.6	97.0$\pm$0.7	81.9$\pm$1.2	78.2$\pm$1.1	58.3$\pm$3.1
	PHP	**91.6$\pm$1.4***	99.2$\pm$0.4	97.6$\pm$0.7	83.4$\pm$1.2	83.2$\pm$1.0	59.1$\pm$3.1
	CoT+SC	91.1$\pm$1.4	99.0$\pm$0.4	97.6$\pm$0.7	85.1$\pm$1.1	83.2$\pm$1.0	69.3$\pm$2.9
	PHP+SC	90.6$\pm$1.5	98.8$\pm$0.4	97.4$\pm$0.7	83.3$\pm$1.2	85.2$\pm$1.0	64.2$\pm$3.0
	Self-Refine	-	-	-	-	75.1	-
	CRITIC	-	-	-	83.3	78.2	-
	Self-Convinced	79.3	-	-	84.9	81.5	62.0
	Multi-Agent (Debate)	-	-	-	-	85.0$\pm$3.5	-
	ReConcile	-	-	-	-	85.3$\pm$2.2	66.0$\pm$0.8
	Self-Verification	90.4	97.4	92.9	83.1	74.9	60.6
	FOBAR	89.4	99.3	94.5	**88.9**	85.1	62.6
	CoT+RAD	**91.6$\pm$1.4***	**99.7$\pm$0.2***	98.0$\pm$0.6	86.2$\pm$1.1	87.5$\pm$0.9	**70.5$\pm$2.9***
	PHP+RAD	91.4$\pm$1.4	99.3$\pm$0.3	**98.4$\pm$0.5***	85.9$\pm$1.1	**88.6$\pm$0.9***	**70.5$\pm$2.9***
GPT-4 Turbo	CoT	**96.5$\pm$0.9***	98.3$\pm$0.5	96.5$\pm$0.8	92.3$\pm$0.8	86.4$\pm$0.9	83.9$\pm$2.3
	PHP	**96.5$\pm$0.9***	98.5$\pm$0.5	97.4$\pm$0.7	93.3$\pm$0.8	91.4$\pm$0.8	83.9$\pm$2.3
	CoT+SC	96.2$\pm$1.0	**98.8$\pm$0.4***	97.0$\pm$0.8	93.4$\pm$0.8	88.5$\pm$0.9	**85.8$\pm$2.2***
	PHP+SC	95.9$\pm$1.0	**98.8$\pm$0.4***	96.9$\pm$0.8	93.9$\pm$0.8	91.1$\pm$0.8	82.7$\pm$2.3
	CoT+RAD	**96.5$\pm$0.9***	**98.8$\pm$0.4***	**98.6$\pm$0.5***	**94.6$\pm$0.7***	**94.6$\pm$0.6***	84.3$\pm$2.3
GPT-4o-mini	CoT	92.9$\pm$1.3	98.8$\pm$0.4	94.5$\pm$1.0	93.5$\pm$0.8	91.5$\pm$0.8	78.7$\pm$2.5
	PHP	93.9$\pm$1.2	98.8$\pm$0.4	95.3$\pm$0.9	93.6$\pm$0.8	93.2$\pm$0.7	78.7$\pm$2.6
	CoT+SC	92.9$\pm$1.3	98.8$\pm$0.4	95.1$\pm$1.0	94.0$\pm$0.8	93.6$\pm$0.7	82.7$\pm$2.4
	PHP+SC	92.9$\pm$1.3	98.8$\pm$0.4	95.1$\pm$1.0	93.4$\pm$0.8	93.4$\pm$0.7	84.3$\pm$2.3
	CoT+RAD	94.4$\pm$1.2	98.8$\pm$0.4	95.7$\pm$0.9	94.1$\pm$0.7	**94.3$\pm$0.6***	84.6$\pm$2.3
	PHP+RAD	**96.5$\pm$0.9***	98.8$\pm$0.4	**98.4$\pm$0.6***	**94.3$\pm$0.7***	**94.3$\pm$0.6***	**85.0$\pm$2.2***

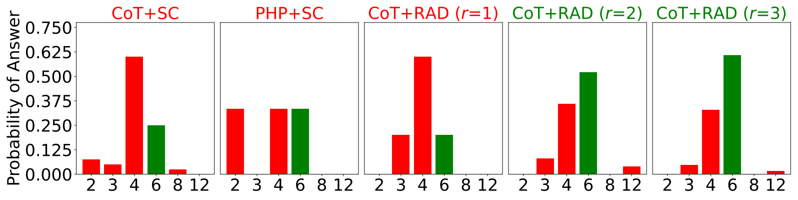


Figure 2: The estimated probabilities of different answers from CoT+SC, PHP+SC, and CoT+RAD (using GPT-3.5 Turbo) for an example from GSM8K dataset.
Question: The ice cream parlor was offering a deal, buy 2 scoops of ice cream, get 1 scoop free. Each scoop cost \$1.50. If Erin had \$6.00, how many scoops of ice cream should she buy? **Answer:** 6.

Figure 2 shows the estimated probabilities of different answers of an example question from GSM8K for all sampling based algorithms using GPT-3.5 Turbo. We observe that, while both CoT+SC and PHP+SC fail to reason correctly, the proposed CoT+RAD outputs the correct answer at both r=2 and 3, although its initial distribution (computed using CoT+SC with B_1=5 samples) does not have a mode at the correct answer. More interestingly, CoT+SC cannot fix the error even if the budget increases to 40 from 5. On the contrary, the proposed CoT+RAD utilizes the additional inference cost effectively to increase the probability of the correct answer at each iteration, demonstrating the usefulness of performing RAD in multiple iterations.

While Figure 2 shows that CoT+RAD has a higher probability of the correct answer for a specific example question, a dataset-level investigation is necessary to determine whether this phenomenon is general. To that end, we restrict ourselves to only the 'difficult' questions in these benchmarks. If a question is correctly solved by all algorithms in Table 1, we categorize it as 'easy'. A question that is not 'easy' is termed 'difficult'. All easy questions are subsequently removed from the datasets[4]. For all 'difficult' questions, we rank CoT+SC, PHP+SC, and CoT+RAD in terms of the probability they assign to the correct answer. The stacked-histograms of these ranks for all six datasets using GPT-4o-mini are shown in Figure 3. We observe that the proposed CoT+RAD achieves the lowest

[4]Since in each of these datasets, the majority of the questions are 'easy', all of CoT+SC, PHP+SC, and CoT+RAD methods assign a very high probability on the correct answers for them. In order to bring out the differences among these algorithms, we only focus on the 'difficult' questions.

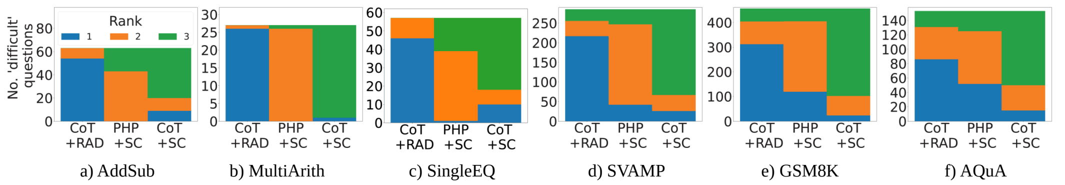


Figure 3: Histogram of **ranks** of the algorithms (the **highest probability of the correct answer** results in the **lowest rank**) for the **'difficult'** questions from all six arithmetic datasets using **GPT-4o-mini**.

rank based on the probability of correct answer across all 'difficult' questions for all datasets more often, outperforming both CoT+SC and PHP+SC. This demonstrates that CoT+RAD has higher probability of the correct answer compared to its competitors for most of these 'difficult' questions, which supports our intuition, presented in Section 3.1. Similar results are obtained for the other two LLMs (see Appendix E).

Table 2: Mean and standard error of accuracy (in %) of reasoning on the MATH dataset using GPT-4o-mini. The **highest** accuracy among all competing algorithms is marked in **bold** and the second-best accuracy in those cases is marked with an underline. The **highest** accuracy is marked with an asterisk if the difference from the second-best accuracy is statistically significant.

Algorithm	Algebra	Counting and Probability	Geometry	Intermediate Algebra	Number Theory	Prealgebra	Precalculus
CoT	88.5±0.9	73.4±2.0	55.1±2.3	51.5±1.6	76.3±1.8	86.9±1.1	49.1±2.1
PHP	90.2±0.9	75.3±2.0	55.9±2.3	52.3±1.7	78.1±1.8	87.6±1.1	51.1±2.1
MACM	90.8±0.9	76.4±2.0	57.4±2.3	55.5±1.7	81.9±1.7	87.8±1.0	51.3±2.1
CoT+SC	93.9±0.7	**82.9±1.7***	64.7±2.2	58.1±1.7	83.5±1.6	**91.2±1.0***	51.3±2.1
PHP+RAD	**94.8±0.6***	80.6±1.8	**65.3±2.2***	**58.9±1.6***	**85.4±1.5***	90.7±1.0	**52.0±2.1***

Results on the MATH Dataset: Table 2 summarizes the experimental results for the MATH dataset, which is a large collection of significantly challenging mathematical reasoning problems. For several sub-disciplines (Geometry, Intermediate Algebra, Precalculus), the state-of-the-art performance (without using extreme computation and a very long inference time) is in the range of 50-65 percent, which suggests that LLMs still find these problems very difficult to solve. Since PHP outperforms CoT for all subcategories, we only evaluate PHP+RAD on these datasets to reduce the token cost, anticipating that PHP+SC would provide better initialization for RAD compared to CoT+SC. Using GPT-4o-mini, the API cost of proposed PHP+RAD is approximately 2.9 cents on average, which is a modest increase from 2.5 cents of CoT+SC. On the contrary, MACM incurs a significantly increased cost of approximately 6.4 cents, due to repeated LLM calls to perform and verify each step and utilization of code-interpreter.

We observe that despite performing an extensive segmentation of the reasoning task and code-based verification of each step, MACM has significantly lower accuracy compared to CoT+SC, which demonstrates that sophisticated prompting approaches often fail to outperform much simpler techniques in a fair experimental setting. The proposed PHP+RAD algorithm leads to a performance improvement in 5 out of 7 settings.

Big-Bench Hard Tasks: In Table 3, we provide results for Date Understanding and Object Tracking, which are problems sets involving quantitative (but not strictly mathematical or arithmetic) reasoning. We observe that PHP still outperforms CoT, demonstrating the utility of refinement via hinting beyond arithmetic tasks. The proposed CoT+RAD offers an improvement in accuracy over the baselines for both of these datasets.

Table 3: Mean and standard error of accuracy (in %) of reasoning for Date Understanding and Object Tracking tasks using GPT-4o-mini. The **highest** accuracy among all competing algorithms is marked in **bold** and the second-best accuracy in those cases is marked with an underline.

Algorithm	Date Understanding	Object Tracking
CoT	91.9±1.4	96.4±0.7
PHP	93.5±1.3	97.7±0.5
CoT+SC	93.8±1.3	96.7±0.7
CoT+RAD	**94.6±1.2***	**98.0±0.5**

5 Related Work

Our proposed method can be situated within a larger literature that aims to improve LLMs' reasoning ability through iterative refinement of chains-of-thought. These works primarily differ in the strategy used to refine the reasoning.

One strand of work involves attempting to iteratively improve single answers, rather than whole distributions of answers like in our work. Progressive Hint Prompting (PHP) (Zheng et al., 2023) proposes to repeatedly generate chains-of-thought, each time encouraging new answers to look like previous answers by providing them as 'hints' to the LLM. Similar works use the same process but push answers away, rather than closer, to the previous answer. Progressive Rectification Prompting (Wu et al., 2024) uses a prompt of the form 'The answer is likely not <hint>', whereas Deliberate-then-Generate (Li et al., 2023) assumes an error was committed and asks the LLM to identify and correct the mistake. Hint-before-Solving Prompting (Fu et al., 2024) also utilizes hints, but in the form of key ideas like a mathematical formula, rather than an answer value.

Instead of trying to improve answers through hints, several works have instead tried to do the same using verbal criticism, at the cost of increased complexity. Self-Refine (Madaan et al., 2023) incorporates a prompt where the LLM self-criticizes its answer, before being queried again with this reflection. Generative Agents (Park et al., 2023) use a similar procedure, albeit in the context of an agent interacting with an environment. CRITIC (Gou et al., 2023) is a more general framework, where the criticism prompt can make use of external tools like a web search engine to offer grounded corrections. Self-Convinced Prompting (Zhang et al., 2023a) and Reflexion (Shinn et al., 2023) expand on Self-Refine by adding extra modules such as a separate answer encoder, or separating the evaluation and self-reflection dimensions of criticism into separate modules. Finally, other related approaches include multi-round debate (Du et al., 2023) and consensus via weighted voting mechanism (Chih-Yao Chen et al., 2023).

Recent studies have, however, cast doubt on the ability of LLMs to self-criticize effectively (Huang et al., 2024; Tyen et al., 2023), leading researchers to consider using a separately trained LLM as the critic. In general, these methods generate a sequence of chains-of-thought, whereas we propose to refine the *distribution* of answers. REFINER (Paul et al., 2023) fine-tunes a separate critic by supervised learning on examples perturbed by hand-designed rules and GPT-3.5 Turbo. Retroformer (Yao et al., 2023) and RL4F (Akyürek et al., 2023) consider fine-tuning of the critic using reinforcement learning instead, which allows for a more precise alignment with the task of improving answers.

Finally, our work can be seen within the greater context of trying to improve chain-of-thought reasoning within large language models. In existing work, several directions for improving CoTs are considered, including construction of better prompts to aid the LLM in reasoning (Fu et al., 2023; Zhang et al., 2023b), fine-tuning with CoTs (Zelikman et al., 2022) so that the LLMs learn to reason, and effective exploration strategies for multi-hop reasoning (Besta et al., 2023; Yao et al., 2023). A recent survey by Chu et al. (2023) provides a comprehensive overview of these techniques. Our contribution is orthogonal to these prompting techniques since we consider improving the *distribution* of answers iteratively rather than focusing on individual CoTs. Novel variants of RAD can be constructed by using these methods for initialization.

6 CONCLUSION

This work presents a novel algorithmic approach, Refined Answer Distributions, to enable an LLM to solve a reasoning task by iteratively refining its inference distribution. The proposed algorithm addresses the issue of the diminishing marginal utility of extra LLM calls for Self-Consistency. RAD focuses on the distribution over the answers at each stage and assigns weights to the previous answers accordingly, concentrating on promising candidates. The marginalization procedure improves sample efficiency. The experimental results, over a range of quantitative reasoning benchmarks and several LLM variants, provide strong evidence that the approach leads to improved reasoning for the same budget of LLM calls, compared to Self-Consistency and other state-of-the-art refinement approaches.

The work could be extended in several directions. Our experiments focus on quantitative reasoning tasks, but the method applies to other types of tasks as long as an appropriate answer refinement strategy would be chosen. For example, in tasks that require a verbal response, the prompt could incorporate 'verbal criticism', based on one of the approaches detailed in Section 5. In addition, in the current version of the procedure, we assign the same number of LLM calls to each unique answer from the previous round. Investigating more efficient strategies to allocate LLM calls non-uniformly to different answers could be another worthwhile direction.

References

P. Aggarwal, A. Madaan, Y. Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proc. Conf. Empirical Methods in Natural Language Process.*, 2023.

A. F. Akyürek, E. Akyürek, A. Madaan, A. Kalyan, P. Clark, D. Wijaya, and N. Tandon. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. *arXiv preprint arXiv:2305.08844*, 2023.

M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, and T. Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *arXiv e-prints arXiv:2308.09687*, 2023.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

J. Chih-Yao Chen, S. Saha, and M. Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. *arXiv e-prints, arXiv:2309.13007*, 2023.

Z. Chu, J. Chen, Q. Chen, W. Yu, T. He, H. Wang, W. Peng, M. Liu, B. Qin, and T. Liu. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arxiv:2309.15402*, 2023.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv e-prints, arXiv:2305.14325*, 2023.

J. Fu, S. Huangfu, H. Yan, S.-K. Ng, and X. Qiu. Hint-before-solving prompting: Guiding LLMs to effectively utilize encoded knowledge. *arXiv preprint arXiv:2402.14310*, 2024.

Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot. Complexity-based prompting for multi-step reasoning. In *Proc. Int. Conf. Learn. Representations*, 2023.

Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. Canton Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. Arrieta Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. Vasuden Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. Singh Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan,

R. Ganapathy, R. Calderer, R. Silveira Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang, S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, and T. Speckbacher. The Llama 3 herd of models. *arXiv e-prints, arXiv:2407.21783*, 2024.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proc. Adv. Neural Inf. Process. Syst.*, 2021.

M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proc. Conf. Empirical Methods in Natural Language Process.*, 2014.

J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou. Large language models cannot self-correct reasoning yet. In *Proc. Int. Conf. Learn. Representations*, 2024.

W. Jiang, H. Shi, L. Yu, Z. Liu, Y. Zhang, Z. Li, and J. T. Kwok. Forward-backward reasoning in large language models for mathematical verification. In *Proc. Findings of Annual Meeting of the Assoc. Comput. Linguist.*, 2024.

T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 22199–22213, 2022.

R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang. Parsing algebraic word problems into equations. *Trans. Assoc. Comput. Linguist.*, pages 585–597, 2015.

B. Lei, Y. Zhang, S. Zuo, A. Payani, and C. Ding. MACM: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. In *Proc. Adv. Neural Inf. Process. Syst.*, 2024.

B. Li, R. Wang, J. Guo, K. Song, X. Tan, H. Hassan, A. Menezes, T. Xiao, J. Bian, and J. Zhu. Deliberate then generate: Enhanced prompting framework for text generation. *arXiv preprint arXiv:2305.19835*, 2023.

W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proc. Conf. Empirical Methods in Natural Language Process.*, 2017.

A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Proc. Adv. Neural Inf. Process. Syst.*, 2023.

OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan,

J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024.

J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proc. ACM Symp. User Interface Software and Technology*, 2023.

A. Patel, S. Bhattamishra, and N. Goyal. Are NLP models really able to solve simple math word problems? In *Proc. Conf. North Amer. Chapter of the Associ. Comput. Linguist.: Human Language Technologies*, 2021.

D. Paul, M. Ismayilzada, M. Peyrard, B. Borges, A. Bosselut, R. West, and B. Faltings. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*, 2023.

S. Roy and D. Roth. Solving general arithmetic word problems. In *Proc. Conf. Empirical Methods in Natural Language Process.*, 2015.

A. Saparov and H. He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. In *Proc. Adv. Neural Inf. Process. Syst.*, 2023.

A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, A. Agarwal, A. Power, A. Ray, A. Warstadt, A. W. Kocurek, A. Safaya, A. Tazarv, A. Xiang, A. Parrish, A. Nie, A. Hussain, A. Askell, A. Dsouza, A. Slone, A. Rahane, A. S. Iyer, A. J. Andreassen, A. Madotto, A. Santilli, A. Stuhlmüller, A. M. Dai, A. La, A. K. Lampinen, A. Zou, A. Jiang, A. Chen, A. Vuong, A. Gupta, A. Gottardi, A. Norelli, A. Venkatesh, A. Gholamidavoodi, A. Tabassum, A. Menezes, A. Kirubarajan, A. Mullokandov, A. Sabharwal, A. Herrick, A. Efrat, A. Erdem, A. Karakaş, B. R. Roberts, B. S. Loe, B. Zoph, B. Bojanowski, B. Özyurt, B. Hedayatnia, B. Neyshabur, B. Inden, B. Stein, B. Ekmekci, B. Y. Lin, B. Howald, B. Orinion, C. Diao, C. Dour, C. Stinson, C. Argueta, C. Ferri, C. Singh, C. Rathkopf, C. Meng, C. Baral, C. Wu, C. Callison-Burch, C. Waites, C. Voigt, C. D. Manning, C. Potts, C. Ramirez, C. E. Rivera, C. Siro, C. Raffel, C. Ashcraft, C. Garbacea, D. Sileo, D. Garrette, D. Hendrycks, D. Kilman, D. Roth, C. D. Freeman, D. Khashabi, D. Levy, D. M. González, D. Perszyk, D. Hernandez, D. Chen, D. Ippolito, D. Gilboa, D. Dohan, D. Drakard, D. Jurgens, D. Datta, D. Ganguli, D. Emelin, D. Kleyko, D. Yuret, D. Chen, D. Tam, D. Hupkes, D. Misra, D. Buzan, D. C. Mollo, D. Yang, D.-H. Lee, D. Schrader, E. Shutova, E. D. Cubuk, E. Segal, E. Hagerman, E. Barnes, E. Donoway, E. Pavlick, E. Rodolà, E. Lam, E. Chu, E. Tang, E. Erdem, E. Chang, E. A. Chi, E. Dyer, E. Jerzak, E. Kim, E. E. Manyasi, E. Zheltonozhskii, F. Xia, F. Siar, F. Martínez-Plumed, F. Happé, F. Chollet, F. Rong, G. Mishra, G. I. Winata, G. de Melo, G. Kruszewski, G. Parascandolo, G. Mariani, G. X. Wang, G. Jaimovitch-Lopez, G. Betz, G. Gur-Ari, H. Galijasevic, H. Kim, H. Rashkin, H. Hajishirzi, H. Mehta, H. Bogar, H. F. A. Shevlin, H. Schuetze, H. Yakura, H. Zhang, H. M. Wong, I. Ng, I. Noble, J. Jumelet, J. Geissinger, J. Kernion, J. Hilton, J. Lee, J. F. Fisac, J. B. Simon, J. Koppel, J. Zheng, J. Zou, J. Kocon, J. Thompson, J. Wingfield, J. Kaplan, J. Radom, J. Sohl-Dickstein, J. Phang, J. Wei, J. Yosinski, J. Novikova, J. Bosscher, J. Marsh, J. Kim, J. Taal, J. Engel, J. Alabi, J. Xu, J. Song,

J. Tang, J. Waweru, J. Burden, J. Miller, J. U. Balis, J. Batchelder, J. Berant, J. Frohberg, J. Rozen, J. Hernandez-Orallo, J. Boudeman, J. Guerr, J. Jones, J. B. Tenenbaum, J. S. Rule, J. Chua, K. Kanclerz, K. Livescu, K. Krauth, K. Gopalakrishnan, K. Ignatyeva, K. Markert, K. Dhole, K. Gimpel, K. Omondi, K. W. Mathewson, K. Chiafullo, K. Shkaruta, K. Shridhar, K. McDonell, K. Richardson, L. Reynolds, L. Gao, L. Zhang, L. Dugan, L. Qin, L. Contreras-Ochando, L.-P. Morency, L. Moschella, L. Lam, L. Noble, L. Schmidt, L. He, L. Oliveros-Colón, L. Metz, L. K. Senel, M. Bosma, M. Sap, M. T. Hoeve, M. Farooqi, M. Faruqui, M. Mazeika, M. Baturan, M. Marelli, M. Maru, M. J. Ramirez-Quintana, M. Tolkiehn, M. Giulianelli, M. Lewis, M. Potthast, M. L. Leavitt, M. Hagen, M. Schubert, M. O. Baitemirova, M. Arnaud, M. McElrath, M. A. Yee, M. Cohen, M. Gu, M. Ivanitskiy, M. Starritt, M. Strube, M. Swedrowski, M. Bevilacqua, M. Yasunaga, M. Kale, M. Cain, M. Xu, M. Suzgun, M. Walker, M. Tiwari, M. Bansal, M. Aminnaseri, M. Geva, M. Gheini, M. V. T, N. Peng, N. A. Chi, N. Lee, N. G.-A. Krakover, N. Cameron, N. Roberts, N. Doiron, N. Martinez, N. Nangia, N. Deckers, N. Muennighoff, N. S. Keskar, N. S. Iyer, N. Constant, N. Fiedel, N. Wen, O. Zhang, O. Agha, O. Elbaghdadi, O. Levy, O. Evans, P. A. M. Casares, P. Doshi, P. Fung, P. P. Liang, P. Vicol, P. Alipoormolabashi, P. Liao, P. Liang, P. W. Chang, P. Eckersley, P. M. Htut, P. Hwang, P. Miłkowski, P. Patil, P. Pezeshkpour, P. Oli, Q. Mei, Q. Lyu, Q. Chen, R. Banjade, R. E. Rudolph, R. Gabriel, R. Habacker, R. Risco, R. Millière, R. Garg, R. Barnes, R. A. Saurous, R. Arakawa, R. Raymaekers, R. Frank, R. Sikand, R. Novak, R. Sitelew, R. L. Bras, R. Liu, R. Jacobs, R. Zhang, R. Salakhutdinov, R. A. Chi, S. R. Lee, R. Stovall, R. Teehan, R. Yang, S. Singh, S. M. Mohammad, S. Anand, S. Dillavou, S. Shleifer, S. Wiseman, S. Gruetter, S. R. Bowman, S. S. Schoenholz, S. Han, S. Kwatra, S. A. Rous, S. Ghazarian, S. Ghosh, S. Casey, S. Bischoff, S. Gehrmann, S. Schuster, S. Sadeghi, S. Hamdan, S. Zhou, S. Srivastava, S. Shi, S. Singh, S. Asaadi, S. S. Gu, S. Pachchigar, S. Toshniwal, S. Upadhyay, S. S. Debnath, S. Shakeri, S. Thormeyer, S. Melzi, S. Reddy, S. P. Makini, S.-H. Lee, S. Torene, S. Hatwar, S. Dehaene, S. Divic, S. Ermon, S. Biderman, S. Lin, S. Prasad, S. Piantadosi, S. Shieber, S. Misherghi, S. Kiritchenko, S. Mishra, T. Linzen, T. Schuster, T. Li, T. Yu, T. Ali, T. Hashimoto, T.-L. Wu, T. Desbordes, T. Rothschild, T. Phan, T. Wang, T. Nkinyili, T. Schick, T. Kornev, T. Tunduny, T. Gerstenberg, T. Chang, T. Neeraj, T. Khot, T. Shultz, U. Shaham, V. Misra, V. Demberg, V. Nyamai, V. Raunak, V. V. Ramasesh, vinay uday prabhu, V. Padmakumar, V. Srikumar, W. Fedus, W. Saunders, W. Zhang, W. Vossen, X. Ren, X. Tong, X. Zhao, X. Wu, X. Shen, Y. Yaghoobzadeh, Y. Lakretz, Y. Song, Y. Bahri, Y. Choi, Y. Yang, S. Hao, Y. Chen, Y. Belinkov, Y. Hou, Y. Hou, Y. Bai, Z. Seid, Z. Zhao, Z. Wang, Z. J. Wang, Z. Wang, and Z. Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Proc. ACL Findings*, 2023.

G. Tyen, H. Mansoor, P. Chen, T. Mak, and V. Cărbune. LLMs cannot find reasoning errors, but can correct them! *arXiv preprint arXiv:2311.08516*, 2023.

X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proc. Int. Conf. Learn. Representations*, 2023.

J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Proc. Adv. Neural Inf. Process. Syst.*, 2022b.

Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, and J. Zhao. Large language models are better reasoners with self-verification. In *Conf. Empirical Methods in Natural Language Process.*, 2023.

Z. Wu, M. Jiang, and C. Shen. Get an A in math: Progressive rectification prompting. In *Proc. AAAI Conf. Artif. Intell.*, 2024.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv e-prints, arXiv:2305.10601*, 2023.

W. Yao, S. Heinecke, J. C. Niebles, Z. Liu, Y. Feng, L. Xue, R. Murthy, Z. Chen, J. Zhang, D. Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*, 2023.

E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman. STaR: Bootstrapping Reasoning With Reasoning. *arXiv e-prints, arXiv:2203.14465*, 2022.

H. Zhang, M. Cai, X. Zhang, C. J. Zhang, R. Mao, and K. Wu. Self-convinced prompting: Few-shot question answering with repeated introspection. *arXiv preprint arXiv:2310.05035*, 2023a.

Z. Zhang, A. Zhang, M. Li, and A. Smola. Automatic chain of thought prompting in large language models. In *Proc. Int. Conf. Learn. Representations*, 2023b.

C. Zheng, Z. Liu, E. Xie, Z. Li, and Y. Li. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*, 2023.

A Pseudocode of RAD

Algorithm 1 Refined Answer Distribution (RAD)

```
1:  Input: task $x$
2:  Hyperparameters: sampling budget $B_{max} > 0$, number of iterations $R > 1$ and $\{B_r > 0\}_{r=1}^{R}$
    such that $\sum_{r=1}^{R} B_r = B_{max}$
3:  Output: answer $\hat{y}$, approximations of $\{p_r(\tilde{y}|x)\}_{r=1}^{R}$
4:  for $r = 0 : R - 1$ do
5:      if $r = 0$ then
6:          Sample $B_1$ answers from $p_1(\cdot|x)$.
7:          Approximate $p_1(\tilde{y}|x)$ using eq. 2.
8:      else
9:          for $m = 1 : M$ do
10:             Sample $\lfloor \frac{B_{r+1}}{M} \rfloor$ answers from $p(\cdot|x, \mathrm{Refine}(y^m))$ in order to form a Monte carlo approx-
                imation, as shown in eq. 3.
11:         end for
12:         Approximate $p_{r+1}(\tilde{y}|x)$ using eqs. 4 and 5.
13:     end if
14:     Find the mode of the approximated $p_R(\tilde{y}|x)$ and assign it to $\hat{y}$.
15: end for
```

B Description of the Benchmark Datasets

We evaluate on the test sets of six arithmetic reasoning benchmarks. Two datasets include simpler problems that can be solved mostly in a single step: AddSub (Hosseini et al., 2014) consists of 395 math word problems that require addition and / or subtraction for the solution, while SingleEQ (Koncel-Kedziorski et al., 2015) contains 508 questions which can be solved using a single equation. Four more challenging datasets require multi-step reasoning: MultiArith (Roy and Roth, 2015) (600 math problems), SVAMP (Patel et al., 2021) (1000 varied math problems), GSM8K (Cobbe et al., 2021) (1319 grade-school level problems), and AQuA (Ling et al., 2017) (254 algebraic word problems). Although these arithmetic problems in the previous benchmarks are relatively simple for humans, LLMs often struggle in solving these types of problems (Patel et al., 2021). In addition, we also conduct experiments on considerably harder MATH (Hendrycks et al., 2021) dataset which contains 5000 competition-level mathematics problems written in LaTeX and natural language. BIG-Bench Hard (Suzgun et al., 2023) consists of 23 difficult tasks from the BIG-Bench suite (Srivastava et al., 2023), where previous large language models did not surpass the average human performance. We focus on the “Date Understanding” and “Object Tracking” tasks, which require quantitative

Table 4: Mean and standard error of accuracy (in %) of few-shot arithmetic reasoning. The **highest** accuracy among all competing algorithms using the same LLM is marked in **bold** and is shown in **red** and **blue** for **Llama-3-8b-instruct** and **Llama-3-70b-instruct** respectively. The second-best accuracy in those cases is marked with an underline and is shown in light red and light blue respectively.

LLM	Algorithm	AddSub	MultiArith	SingleEQ	SVAMP	GSM8K	AQuA
Llama-3-8b-instruct	**CoT**	88.9±1.6	96.7±0.7	90.0±1.3	83.5±1.2	76.6±1.2	51.2±3.1
	PHP	90.4±1.5	94.7±0.9	91.1±1.3	86.4±1.1	76.8±1.2	57.1±3.1
	CoT+SC	91.1±1.4	**98.0±0.6***	94.5±1.0	**90.4±0.9***	**85.0±1.0***	59.4±3.1
	CoT+RAD	**92.9±1.3***	96.8±0.7	94.9±1.0	90.1±0.9	82.3±1.1	60.0±3.0
	PHP+RAD	**92.9±1.3***	97.8±0.6	**95.1±1.0***	**90.4±0.9***	84.2±1.1	**66.1±3.0***
Llama-3-70b-intruct	**CoT**	-	-	-	91.2±0.9	93.2±0.7	72.8±2.8
	PHP	-	-	-	91.9±0.9	93.3±0.7	73.2±2.8
	CoT+SC	-	-	-	92.6±0.8	94.2±0.6	78.0±2.6
	CoT+RAD	-	-	-	**93.1±0.8***	94.2±0.6	**79.9±2.5***
	PHP+RAD	-	-	-	92.7±0.8	**94.6±0.6***	78.7±2.6

reasoning. Answering questions from the Date Understanding dataset involves inferring a date from a given scenario. Object tracking task evaluates an algorithm's ability to reason and determine the final state of objects, after applying a sequence of shuffling, starting from their known initial states. All of these benchmarks are available under open-source licenses [5].

C Experimental Results using Llama Models

We have conducted experiments with two Llama-family LLMs: the weaker Llama-3-8b-instruct and the very capable Llama-3-70b-instruct. In order to reduce the API cost of the experiments, we restrict running the more expensive 70B model to only the three most difficult benchmarks.

From the results in Table 4, we observe that using Llama-3-8b-instruct, the relative advantage of PHP over CoT is diminished in comparison to the GPT models. This suggests that weaker LLMs, such as Llama-3-8b-instruct, which often have relatively poor instruction following capability, cannot utilize the hint effectively for solving the reasoning task, highlighting the inadequacy of sophisticated prompting for weaker LLMs. In this setting, the effect of the quality of approximation of the initial distribution of RAD becomes important for obtaining a good reasoning accuracy and PHP+RAD outperforms CoT+RAD in most cases. Except for GSM8K, PHP+RAD either outperforms CoT+SC or obtains comparable performance on all other datasets. On the contrary, for a strongly capable Llama-3-70b-instruct model, both CoT+RAD and PHP+RAD perform well.

D Results for the 'Difficult' Questions

In order to demonstrate the advantage of CoT+RAD more clearly, we restrict ourselves to only the 'difficult' questions in the six arithmetic benchmarks. If a question is solved correctly by all algorithms in Table 5, we categorize it as 'easy'. A question which is not 'easy' is termed 'difficult'. All easy questions are subsequently removed from the datasets to compute the accuracies only on the difficult questions. From Table 5, we observe that the relative accuracy gains offered by the proposed CoT+RAD algorithm are more substantial in most cases.

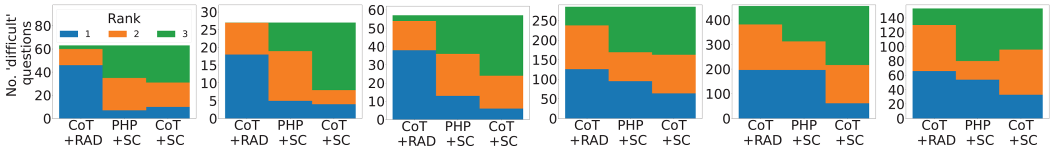
E Additional Results for Comparing Probability of Correct Answer

Figure 3 in the main paper shows that in comparison to CoT+SC and PHP+SC using GPT-4o-mini, CoT+RAD assigns higher probability to the correct answers for most of the 'difficult' questions across all datasets. Figures 4 and 5 demonstrate that the same trend holds for both GPT-3.5-Turbo and GPT-4-Turbo LLMs.

[5] CC-BY-4.0 [AddSub; SingleEQ], Apache 2.0 [MultiArith; AQuA] and MIT [SVAMP; GSM8K; MATH; BIG-Bench Hard].

Table 5: Mean and standard error of accuracy (in %) of few-shot arithmetic reasoning for the 'difficult' questions. The **highest** accuracy among all competing algorithms using the same LLM is marked in **bold** and is shown in **red**, **blue**, and **orange** for **GPT-3.5 Turbo**, **GPT-4 Turbo**, and **GPT-4o-mini** respectively. The second-best accuracy in those cases is marked with an underline and is shown in light red, light blue, and light orange respectively. The **highest** accuracy is marked with an asterisk if the difference from the second-best accuracy is statistically significant.

LLM	Algorithm	AddSub	MultiArith	SingleEQ	SVAMP	GSM8K	AQuA
GPT-3.5 Turbo	CoT	46.0±6.3	51.9±9.6	73.7±5.8	36.5±2.9	37.1±2.2	30.7±3.7
	PHP	**47.6±6.2***	81.5±7.4	78.9±5.4	41.8±2.9	51.3±2.3	32.0±3.7
	CoT+SC	44.4±6.3	77.8±7.9	78.9±5.4	47.7±3.0	51.3±2.3	49.0±4.0
	PHP+SC	41.3±6.3	74.1±8.4	77.2±5.6	41.4±2.9	57.2±2.3	40.5±4.0
	CoT+RAD	**47.6±6.3***	**92.6±5.1***	**82.5±5.1***	**51.6±3.0***	**63.8±2.2***	**51.0±4.0***
GPT-4 Turbo	CoT	**77.8±5.3**	63.0±9.3	68.4±6.2	73.0±2.6	60.7±2.3	73.2±3.6
	PHP	**77.8±5.3**	66.7±9.2	77.2±5.5	76.5±2.5	75.2±2.0	73.2±3.6
	CoT+SC	76.2±5.4	**74.1±8.4***	73.7±5.8	76.8±2.5	66.7±2.2	**76.5±3.5***
	PHP+SC	74.6±5.4	**74.1±8.5***	71.9±5.9	78.6±2.4	74.3±2.1	71.2±3.6
	CoT+RAD	**77.8±5.3**	**74.1±8.4***	**87.7±4.4***	**81.1±2.3***	**84.4±1.7***	73.9±3.5
GPT-4o-mini	CoT	55.6±6.2	74.1±8.5	50.9±6.6	77.2±2.5	75.4±2.0	64.7±3.9
	PHP	61.9±6.2	74.1±8.4	57.9±6.6	77.5±2.5	80.3±1.9	64.7±3.8
	CoT+SC	55.6±6.3	74.1±8.4	56.1±6.6	78.9±2.4	81.6±1.8	71.2±3.7
	PHP+SC	55.6±6.3	74.1±8.5	56.1±6.5	76.8±2.5	80.9±1.9	73.9±3.5
	CoT+RAD	**65.1±6.1***	74.1±8.4	**61.4±6.4***	**79.3±2.4***	**83.6±1.7***	**74.5±3.5***

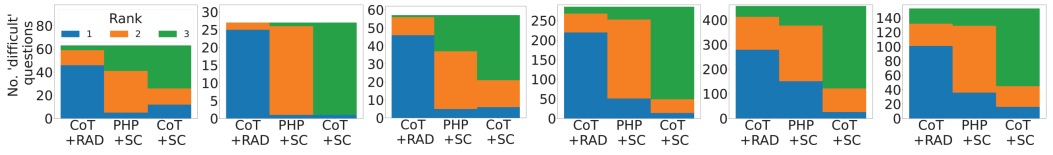


a) AddSub b) MultiArith c) SingleEQ d) SVAMP e) GSM8K f) AQuA

Figure 4: Histogram of **ranks** of the algorithms (the **highest probability of the correct answer** results in the **lowest rank**) for the **'difficult'** questions from all six arithmetic datasets using **GPT-3.5 Turbo**.

In order to demonstrate the statistical significance of the increase in probability of the true answer, we conduct a Wilcoxon signed rank test between $p_3(y|x)$ (i.e., the estimated probability of the true answer obtained from the proposed CoT+RAD) and $p_1(y|x)$ (i.e., the probability of the true answer, at the initialization of CoT+RAD, estimated from CoT+SC using 40 samples), and report the p-values in Table 6. We observe that except for 5 out of 36 cases (6 datasets, 3 LLMs, and 2 different partitions of the datasets), the difference between $p_3(y|x)$ and $p_1(y|x)$ is statistically significant at the 5% level, providing strong empirical support in favor of the capability of the RAD iterations in increasing the probability of the true answers.

In addition, we also calculate the percentage of difficult questions for which $p_3(y|x) \geqslant p_1(y|x)$ is satisfied and report the results in Table 7. We observe that in each case, for the majority of the questions, RAD iterations do not decrease the probability of the true answer.



a) AddSub b) MultiArith c) SingleEQ d) SVAMP e) GSM8K f) AQuA

Figure 5: Histogram of **ranks** of the algorithms (the **highest probability of the correct answer** results in the **lowest rank**) for the **'difficult'** questions from all six arithmetic datasets using **GPT-4 Turbo**.

Table 6: p-value from Wilcoxon signed rank test between the probabilities of true answers from distributions $p_3(y|x)$ and $p_1(y|x)$ for the 'difficult' questions (for the entire dataset)

LLM	AddSub	MultiArith	SingleEQ	SVAMP	GSM8K	AQuA
GPT-3.5 Turbo	0.0291 (0.1172)	0.0006 (1.3×10^{-5})	0.0012 (8.6×10^{-5})	0.0132 (1.4×10^{-5})	9.2×10^{-18} (4.3×10^{-22})	0.0001 (1.6×10^{-8})
GPT-4 Turbo	0.2868 (0.2258)	0.0104 (2.3×10^{-6})	0.0002 (6.2×10^{-7})	4.8×10^{-8} (1.7×10^{-13})	2.2×10^{-31} (1.5×10^{-41})	0.0065 (0.0042)
GPT-4o-mini	0.0038 (0.0024)	0.8413 (0.0243)	0.0317 (0.0255)	0.5898 (0.3028)	4.5×10^{-12} (5.2×10^{-12})	2.1×10^{-5} (8.5×10^{-6})

Table 7: Percentage of 'difficult' questions (percentage of questions in the entire dataset), so that $p_3(y|x) \geqslant p_1(y|x)$ is satisfied (in other words, RAD does not decrease the probability of the true answer)

LLM	AddSub	MultiArith	SingleEQ	SVAMP	GSM8K	AQuA
GPT-3.5 Turbo	79.4 (92.7)	85.2 (97.3)	86.0 (97.2)	63.5 (83.8)	70.8 (81.4)	64.7 (74.8)
GPT-4 Turbo	76.2 (95.7)	96.3 (99.7)	87.7 (98.0)	89.5 (96.9)	85.7 (93.3)	79.1 (86.6)
GPT-4o-mini	85.7 (97.2)	96.3 (99.7)	82.5 (97.0)	81.1 (93.9)	83.8 (92.7)	75.8 (83.9)

F Refinement Prompt using Hinting (Zheng et al., 2023)

A typical arithmetic reasoning question is presented in Table 8, where the chain-of-thought yielded the incorrect answer $y = 21$. In Table 9, the hinting-based refinement prompt of PHP is applied to this same question with the hints $y' = 4, 7$, yielding a chain-of-thought with the correct answer $y = 6$.

Table 8: Example problem from Zheng et al. (2023).

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?

A: There are 21 trees in the Grove today. Grove workers are done planting trees. So 21 trees were planted in the Grove.

Table 9: Demonstration of the hinting prompt from Zheng et al. (2023), as applied to the example problem from Table 8. Additions are highlighted in blue.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today? (Hint: The answer is near to 4, 7).

A: We know the Answer Hints: 4, 7. With the Answer Hints: 4, 7, we will answer the question. There are 15 trees originally. Then there were 21 trees after the Grove workers planted some more. So there must have been 21 - 15 = 6 trees that were planted. The answer is 6.

G Examples of Full Prompts

We provide examples of full prompts for CoT in Table 10, PHP in Table 11 and our method CoT+RAD in Table 12. As CoT+SC and PHP+SC are simply sampled versions of CoT and PHP, the prompts are identical as in those cases. The base examples are taken from Zheng et al. (2023).

Table 10: An example of a CoT prompt with 4 few-shot examples.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally.Then there were 21 trees after the Grove workers planted some more. So there must have been 21 - 15 = 6 trees that were planted. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are originally 3 cars. Then 2 more cars arrive. Now 3 + 2 = 5 cars are in the parking lot. The answer is 5.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Originally, Leah had 32 chocolates and her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39 pieces left in total. The answer is 39.

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason had 20 lollipops originally. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8 lollipops. The answer is 8.

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A:

Table 11: An example of a PHP prompt with 4 few-shot examples. The few-shot questions are provided true answer, while the test question is provided with the history so far, in this case $y' = 7, 11, 8$.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today? (Hint: The answer is near to 6).
A: We know the Answer Hints: 6. With the Answer Hints: 6, we will answer the question. There are 15 trees originally. Then there were 21 trees after the Grove workers planted some more. So there must have been 21 - 15 = 6 trees that were planted. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot? (Hint: The answer is near to 5).
A: We know the Answer Hints: 5. With the Answer Hints: 5, we will answer the question. There are originally 3 cars. Then 2 more cars arrive. Now 3 + 2 = 5 cars are in the parking lot. The answer is 5.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total? (Hint: The answer is near to 39).
A: We know the Answer Hints: 39. With the Answer Hints: 39, we will answer the question. Originally, Leah had 32 chocolates and her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39 pieces left in total. The answer is 39.

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny? (Hint: The answer is near to 8).
A: We know the Answer Hints: 8. With the Answer Hints: 8, we will answer the question. Jason had 20 lollipops originally. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8 lollipops. The answer is 8.

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now? (Hint: The answer is near to 7, 11, 8).
A:

Table 12: An example of a CoT+RAD prompt with 4 few-shot examples. The few-shot questions are provided with true answers as hints, while the test question is provided with one of the distinct answers obtained during the interaction with the LLM in the previous iteration. In this case, $y' = 8$.

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today? (Hint: The answer is near to 6).
A: We know the Answer Hints: 6. With the Answer Hints: 6, we will answer the question. There are 15 trees originally. Then there were 21 trees after the Grove workers planted some more. So there must have been 21 - 15 = 6 trees that were planted. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot? (Hint: The answer is near to 5).
A: We know the Answer Hints: 5. With the Answer Hints: 5, we will answer the question. There are originally 3 cars. Then 2 more cars arrive. Now 3 + 2 = 5 cars are in the parking lot. The answer is 5.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total? (Hint: The answer is near to 39).
A: We know the Answer Hints: 39. With the Answer Hints: 39, we will answer the question. Originally, Leah had 32 chocolates and her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39 pieces left in total. The answer is 39.

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny? (Hint: The answer is near to 8).
A: We know the Answer Hints: 8. With the Answer Hints: 8, we will answer the question. Jason had 20 lollipops originally. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8 lollipops. The answer is 8.

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now? (Hint: The answer is near to 8).
A: