Solving Math Word Problems via Cooperative Reasoning induced Language Models

Xinyu Zhu$^{\diamond *}$ **Junjie Wang**$^{\spadesuit *}$ **Lin Zhang**$^{\heartsuit}$ **Yuxiang Zhang**$^{\spadesuit}$
Yongfeng Huang$^{\clubsuit}$ **Ruyi Gan**$^{\heartsuit}$ **Jiaxing Zhang**$^{\heartsuit}$ **Yujiu Yang**$^{\diamond \dagger}$

$^{\diamond}$Tsinghua University $^{\spadesuit}$Waseda University
$^{\heartsuit}$IDEA $^{\clubsuit}$The Chinese University of Hong Kong

zhuxy21@mails.tsinghua.edu.cn   yang.yujiu@sz.tsinghua.edu.cn

wjj1020181822@toki.waseda.jp   joel0495@asagi.waseda.jp

{zhanglin, ganruyi, zhangjiaxing}@idea.edu.cn

Abstract

Large-scale pre-trained language models (PLMs) bring new opportunities to challenging problems, especially those that need high-level intelligence, such as the math word problem (MWPs). However, directly applying existing PLMs to MWPs can fail as the generation process lacks sufficient supervision and thus lacks fast adaptivity as humans. We notice that human reasoning has a dual reasoning framework that consists of an immediate reaction system (system 1) and a delicate reasoning system (system 2), where the entire reasoning is determined by their interaction. This inspires us to develop a cooperative reasoning-induced PLM for solving MWPs, called **Co**operative **Re**asoning (**CoRe**), resulting in a human-like reasoning architecture with system 1 as the generator and system 2 as the verifier. In our approach, the generator is responsible for generating reasoning paths, and the verifiers are used to supervise the evaluation in order to obtain reliable feedback for the generator. We evaluate our CoRe framework on several mathematical reasoning datasets and achieve decent improvement over state-of-the-art methods, up to 9.6% increase over best baselines.[1]

1 Introduction

Addressing math problems is a hallmark of human intelligence, which allows reasoning and adapting from limited data. We want neural models to be able to do the same, however, quick and flexible reasoning is challenging to current neural models as they must possess a certain level of prior experience from a limited amount of new data while avoiding overfitting. The rapid growth of large-scale Pre-trained Language Models (PLMs) offers
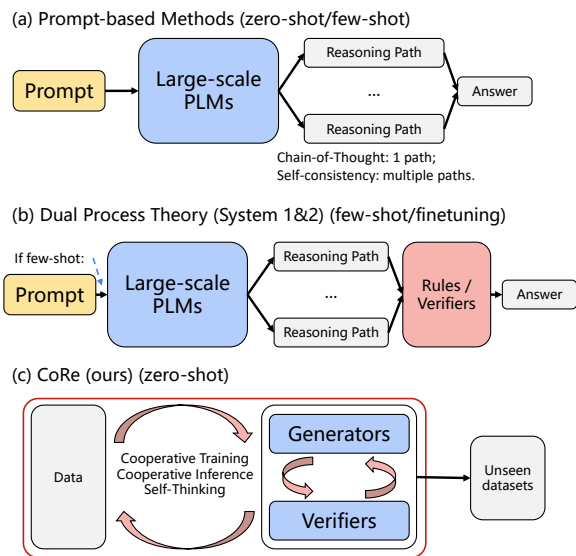

Figure 1: Comparing our CoRe with popular methods in mathematical logic reasoning tasks.

unprecedented potential for this issue, often relying on well-designed trigger prompts (Wei et al., 2022c; Li et al., 2022; Brown et al., 2020). Although appealing in terms of efficiency, its success relies on memorizing patterns with a sufficiently large number of parameters ($\geq$ 100 billion) (Wei et al., 2022b), differentiating it from the fast adaptivity in the human reasoning process.

Active disciplines like neuroscience and cognitive science attempt to uncover the mechanism of human reasoning, and agree that our learning process is governed by an interaction mechanism, often referred to as System 1 and System 2 (Evans, 2003; Kahneman, 2011). In particular, System 1 offers fast responses like human instinct, and System 2 performs deliberate reasoning. Interactions between them are important for adapting to a continuously changing environment. PLMs behave more like System 1, according to the above theory, and thus lack the generalization ability in reasoning (Nye et al., 2021).

---

*Equal contribution.

†Corresponding Author.

[1]Our codes are available at https://github.com/TianHongZXY/CoRe

4471

In this work, we explore a new line of zero-shot math problem reasoning, using a human reasoning-alike framework with feedback in the solution generation loop as opposed to pure PLM-based methods, called **Co**operative **Re**asoning (**CoRe**). Intuitively, System 1 and System 2 are embodied as generators and verifiers, respectively, and they are defined as follows: generators for generating reasoning paths, and verifiers for supervising the paths' evaluation. Specifically, we train a LM beyond the question-answer paradigm by integrating in-the-loop reasoning, i.e., we let the LM output both the answer and the corresponding reasoning process for a given question. Meanwhile, we introduce two types of verifiers, including token-level and sentence-level, allowing us to provide feedback in the whole solution generation life-cycle. Notice that the solution path is generated by selecting candidate tokens with some probability so that it is tree-alike and much coincides with the tree search process of Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006). With this in mind, the verifiers can score tokens along the solution generation process from start to end when using the MCTS. Therefore, we can use the score to evaluate the quality of the generation process during inferring before finalizing the solution, making timely feedback available for supervising the generation process. With this, the evaluation goes beyond the quality of the final result at the granularity of each reasoning step, extending the supervision from the solution level to the path level. We combine the solution score and the perplexity of its corresponding reasoning path to encourage the overall training towards high-quality augmented solutions while aligning with the reliable reasoning process, aiming to improve generalization ability.

Our experimentally evaluate CoRe on multiple mathematical reasoning datasets in both zero-shot and fine-tuning settings. CoRe consistently achieves better performance than competing baselines. Notably, CoRe has up to 9.6% improvements on MultiArith over SoTA baselines, which are dozens of times larger than our model.

In summary, our contributions are as follows.

- We propose a novel reasoning method for mathematical problem solving, called **Co**operative **Re**asoning (**CoRe**), that introduces feedback in the loop during solution generation as opposed to the sequential learning process in the previous ones, resulting in the first method for this task that builds on top of the learning mechanism in the human brain.
- We develop a self-thinking strategy for further boosting reasoning ability with generated data from the cooperation between System 1 and System 2.
- We demonstrate the superiority of CoRe comparing to other zero-shot and fine-tuning methods, which has 9.6% improvements on MultiArith over SoTA baselines.

2 Related Work

2.1 Dual Process System

Dual-process theory (Evans, 2003; Kahneman, 2011) argues there are two cognitive systems underpinning human reasoning: System 1 and System 2. The purpose of clarifying these systems is that they have the potential to help us construct artificial intelligence systems that benefit from human flexibility and methodical generalization.

Dual process system model guidance is not new. Nye et al. (2021) simulated Systems 1 and 2 to improve consistency and coherence of neural networks. Similar to several studies Cobbe et al. (2021); Li et al. (2022); Scialom et al. (2021), in addition to System 1 for the generation, we develop a distinct model as System 2, called Verifier. The Verifier checks the feasibility and correctness of the generator's content and collaboratively solves the reasoning task together.

2.2 Multi-step Reasoning

Many works exploit the multi-step reasoning ability of language models. Cobbe et al. (2021) showed that training a verifier to score the solutions generated by a fine-tuned GPT-3 could improve the performance compared to solely fine-tuning a GPT-3. Nye et al. (2022) discovered that asking the language model to write the intermediate process could achieve better results on various NLP tasks. Likewise, Chain-of-Thought (CoT) prompts (Wei et al., 2022c) prepended exemplars with intermediate reasoning steps as prompts and achieved SoTA on several reasoning benchmarks by using large-scale PLMs. Wang et al. (2022) further boosted CoT's performance by sampling a bunch of possible solutions and then obtained the final answer by majority voting. DIVERSE (Li et al., 2022) proved diverse CoT prompts and an

extra verifier were both helpful for PLMs to solve reasoning problems. Kojima et al. (2022) found that by simply adding "Let's think step by step" after the question. PLMs could successfully step by step solve the problems, called Zero-shot-CoT.

These above methods rely on extremely large language models, resulting in high computational cost and time-consuming. Moreover, several works (Wei et al., 2022c; Kojima et al., 2022) point out that neither CoT nor Zero-shot-CoT is helpful to smaller models. While our method does not necessarily require extremely large PLMs and can work with models with different size scales, thus reducing computational cost and inference time. Our approach has competitive zero-shot performance thanks to the efficient and collaborative application of a dual-process system.

3 Cooperative Reasoning

In this section, we will present the proposed cooperative reasoning framework, CoRe, that enforces System 1 and System 2 mutually cooperating, which includes 3 sequential steps: cooperative training, cooperative inference, and self-thinking.

3.1 Preparation

As discussed in Sec. 1, we expect a PLM (G) to fast generate multiple reasoning paths like System 1. Then, considering that System 2 is responsible for deliberate evaluations of the reasoning paths, we employ two modules: a step verifier (V_{step}) for reasoning steps, and a path verifier (V_{path}) for reasoning paths.

3.2 Cooperative Training

Before applying System 1&2 to inference, a critical issue for them is *learn how to generate reasoning paths and evaluate reasoning steps/paths*. Inspired by a widely-used training strategy for reasoners (Cobbe et al., 2021), we present a cooperative training method as shown in Fig. 2 Step 1. Moreover, we discuss hyper-parameter configurations and extra training details in Appendix B.1 and Appendix B.2.

Step 1.1: We first fine-tune G on a dataset $D = \{(q_i, p_i, gt_i)\}_{i=1}^{N}$ consisting of N samples. Each sample x is composed of a question q, a reasoning path p and a ground truth answer gt. We fine-tuen G with standard language modeling objective

$\mathcal{L}_{LM}$ as Eq. (1).

$$\mathcal{L}_{LM} = - \sum_{i=1}^{|p|+|gt|} \log P\left(x_i \mid x_{<i}\right) \qquad (1)$$

Step 1.2: Once G has learned how to generate solutions, we employ it on questions q from D. As a result, we obtain a new dataset $D^+ = \left\{\left(q_i, rp_{i,j}, a_{i,j}\right)\right\}_{\substack{i=1,...,N \\ j=1,...,M}}$ with M generated reasoning paths (rp) and answers (a) for each q.

Step 1.3: Different from the popular methods, we train two verifiers to model human reasoning procedure with deliberate analysis for each step and the whole path. To evaluate several reasoning steps in a path, we desire a token-level scorer, which is named step verifier V_{step}. Therefore, we fine-tune a PLM with two tasks jointly: 1) the language modeling task mentioned before; 2) the verification task to predict a score for each token in the solution. The verification loss $\mathcal{L}_{VS}$ is calculated as the Mean Squared Error (MSE) of the predicted score with respect to the label as follows:

$$\mathcal{L}_{VS} = \sum_{i=1}^{|rp|+|a|} (score_i - \mathbb{I}(a == gt))^2, \qquad (2)$$

where, (rp, a) from D^+ and gt with same q from D.

On the other hand, we need a path-level scorer for reasoning paths. Different from step verifier, we simply extract an overall presentation of the reasoning path for prediction. Specifically, we employ a BERT-like model and take the [CLS] token to calculate MSE loss $\mathcal{L}_{VP}$ similar to $\mathcal{L}_{VS}$.

In summary, the overall training objective for verifiers is given by:

$$\mathcal{L}_V = \mathcal{L}_{VS} + \mathcal{L}_{LM} + \mathcal{L}_{VP}. \qquad (3)$$

3.3 Cooperative Inference

After obtaining a generator and two verifiers, we propose cooperative inference to generate solutions for unseen questions. Instead of treating verifiers as voters, we argue that verifiers should offer appropriate guidance and feedback during the reasoning process. Therefore, we integrate a cooperative search algorithm. In particular, we adopt the popular Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006) to enable controlled reasoning. The cooperative inference starts from the root node, which preserves question tokens. We detail the cooperative inference process as follows.

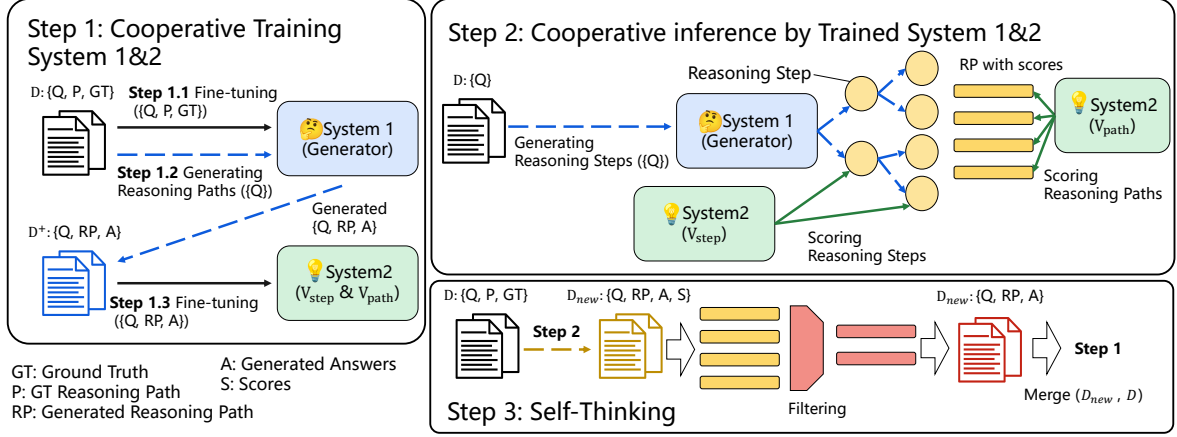**Selection.** If the current node has children, with 50% probability, we select a node from its children

Figure 2: Cooperative reasoning framework.

with the modified PUCT formula (Czech et al., 2021) as Eq. (4),

$$n^* = \arg\max_{n \in \mathcal{C}} (R(n) + c_{puct}\pi(n|s)\frac{\sqrt{\sum_{b \in \mathcal{C}} N(s,b)}}{1 + N(s,n)}), \quad (4)$$

where the state s represents the sequence consisting of all tokens in the current search path. And, $N(s,n)$ means the times that node n has been selected in state s. Reward $R(n)$ records all the scores received from the backup. We perform selection again with the selected node as the current node. Otherwise, we perform expansion once and choose the returned new node as current node.

Expansion. During expansion, the generator is required to generate a sequence of tokens based on the current state. A new node is created to store the generated tokens and added to the current node's children. Then, V_{step} evaluates the current reasoning path and predict a score $score_{step}$. Finally, the new node is returned.

Roll-Out. After selection and expansion, we start from the current node and let the generator complete the reasoning path until it meets [EOS] token or reaches the max token length limit. Next, V_{path} evaluates the whole reasoning path and produces a score $score_{path}$. Remember that V_{step} also provides a score $score_{step}$ during the expansion. Therefore to leverage both scores, we introduce a hyper-parameter α to adjust their contributions to the node's reward,

$$s = score_{path} + \alpha \times score_{step} \quad (5)$$

where s is the final score that each node receives by the backup.

Backup. We update the rewards back from the current node to the root node. The scores produced

---

Algorithm 1 Self-Thinking

Input: Generator G; Step verifier V_{step}; Path verifier V_{path}; Dataset D.
1: Combine generator and verifiers with a cooperative search algorithm.
2: **repeat**
3:   Generate a new dataset D_{new} from input questions.
4:   Filter D_{new}.
5:   Merge D_{new} with D in Step 1.
6:   Do Step 1.
7:   Do Step 2.
8: **until** performance is saturated.

by verifiers are added to $R(n)$ and the visited time $N(s,n)$ is increased by 1.

3.4 Self-Thinking

It is challenging to fine-tune models on the data synthesized by themselves, which indicates they have to be very confident in the content they generate. A proper self-training method can enhance the robustness of the whole system and allow deep data mining. Therefore, we introduce self-thinking as described in Fig. 2 Step 3 and Algorithm 1. Considering the noise contained in generated data, we build a filter by using scores from verifiers and perplexity (PPL) from the generator. In detail, we select high-quality reasoning paths by setting a score threshold. Moreover, we only keep the reasoning paths with no higher PPL than the ground truth solutions. After filtering, we merge D_{new} with D and send it to Step 1. Once the several iterations are completed, we obtain a powerful System 1&2. More details can be found in Appendix B.3.

3.5 Zero-shot Inference

We simply perform cooperative inference as Fig. 2 Step 2 with trained System 1&2 on unseen datasets. After obtaining several reasoning paths with scores, we arrive at the final answer by weighted voting based on scores following (Li et al., 2022).

4 Experiments

4.1 Experimental Setup

4.1.1 Datasets

We consider several widely-used math word problem datasets: GSM8K (Cobbe et al., 2021), ASDiv-A (Miao et al., 2020), SingleOp (Roy et al., 2015), SinlgeEq (Koncel-Kedziorski et al., 2015) and MultiArith (Roy and Roth, 2015). (Details in Appendix A). Following the general setting as in (Kojima et al., 2022; Wei et al., 2022c), we employ accuracy as the evaluation metric for all datasets.

4.1.2 Baselines

For comparison under the zero-shot setting, the results of Instruct GPT-3 (175B) and PaLM (540B) with their various methods are from Kojima et al. (2022). The zero-shot* and zero-shot-CoT* imply not the standard prompt (see details in Appendix B.4). We also provide our generator as a baseline when compared to previous fine-tuning methods. Regarding to sampling multiple solutions, we search 40 paths with the same setting as Self-Consistency (Wang et al., 2022).

For GSM8K, we select various powerful PLMs enhanced by the chain of thought prompt as baselines, including LaMDA (137B) (Thoppilan et al., 2022), GPT-3 (175B) (Brown et al., 2020) and PaLM (540B) (Chowdhery et al., 2022). Except for the few-shot methods, we also include a fine-tuned baseline that applies two GPT-3 (175B), one as the generator and the other as verifier (Cobbe et al., 2021).

4.1.3 Implementation Details

Since cooperative training requires a high-quality dataset with reasoning paths, we treat GSM8K (Cobbe et al., 2021) as the seed dataset D in Sec. 3.2. Unless otherwise, we employ GPT-J (Wang and Komatsuzaki, 2021) as the generator and the step verifier, DeBERTa-large (He et al., 2021) as the path verifier. Since the default setting consists of two GPT-J (6B) and a DeBERTa-

Backbone	Method	SingleEq	MultiArith
Instruct GPT-3 175B	zero-shot	74.6	17.7
	zero-shot*	78.7	22.7
	zero-shot-CoT	78.0	78.7
	zero-shot-CoT*	78.7	79.3
PaLM 540B	zero-shot	-	25.5
	zero-shot-CoT	-	66.1
	+ Self-Consistency	-	89.0
GPT-J 12B	CoRe (**ours**)	**79.5**	**97.5**

Table 1: Zero-shot results on SingleEq and MultiArith.

large (0.4B), we note our backbone as "GPT-J 12B", which implies around 12.4 billion parameters in total. During generation, we apply calculator as assistant following Cobbe et al. (2021). We run all the experiments for 3 times and report the best result, detailed hyper-parameters setting can be found in Appendix B.1. Our zero-shot setting is similar to the transferring setting in T0 (Sanh et al., 2022) and FLAN (Wei et al., 2022a). All the training and testing procedures are done on a DGX station with 8 A100 GPUs.

4.2 Main Results

4.2.1 Zero-shot Results

Table 1 presents main results on two mathematical reasoning datasets, demonstrating the zero-shot generalization ability. CoRe achieves superior performance on both datasets, demonstrating its capability of mathematical reasoning on unseen datasets. Note that the baselines are several dozen times larger than ours and still underperform our model. The improvement might be explained by two potential reasons. One is that applying the CoRe framework on PLMs can activate their reasoning ability, even though their scales are small ($\leq 100B$). Another one is that self-thinking can provide valuable self-produced data to teach Systems 1&2. Therefore, the results present the effectiveness of cooperative working with System 1&2 and self-thinking.

4.2.2 Zero-shot v.s. Fine-tuning

We compare CoRe with previous fine-tuned SoTA baselines on four datasets, and results are presented in Table 2. To show the importance of cooperative reasoning, we apply our generator as a baseline. The results demonstrate that without any guidance generator underperforms previous methods on most datasets. Despite the gain from self-consistency, it still lags behind other fine-tuned SoTAs. While after applying our method CoRe, it sur-

Backbone	Method	ASDiv-A	SingleOp	SingleEq	MultiArith
Fine-tune					
	Previous SoTA	75.3^a	80.1^b	72.3^c	60.5^d
Zero-shot					
GPT-J 6B	Generator only	51.7	53.2	49.2	77.3
	+ Self-Consistency	63.7	59.6	60.2	92.3
GPT-J 12B	CoRe (**ours**)	**90.5**	**85.2**	**79.5**	**97.5**

Table 2: Zero-shot results v.s. previous fine-tuned SoTA results on math reasoning tasks. The previous SoTA baselines are obtained from:a: (Lan et al., 2022), b: LogicForm (Liang et al., 2016), c: UNITDEP (Roy and Roth, 2017), d: Relevance and LCA operation classifier (Roy and Roth, 2015). The best scores are in **bold**.

Backbone	Method	GSM8K
few-shot		
LaMDA 137B	CoT	17.1
	+ Self-Consistency	27.7
GPT-3 175B	CoT	49.6
	+ Self-Consistency	-
PaLM 540B	CoT	56.5
	+ Self-Consistency	**74.4**
fine-tune		
GPT-3 350B	-	57.0
GPT-J 12B	CoRe (**ours**)	<u>63.2</u>

Table 3: Fine-tuning v.s. Few-shot results on GSM8K with various PLMs. Results are reported from Cobbe et al. (2021); Wei et al. (2022c); Wang et al. (2022). The best score is in **bold** and the second is <u>underlined</u>.

Guidance	α	SingleOp	MultiArith
w/o verifiers	-	59.6	92.3
V_{path} only	0	80.2	95.8
$V_{path} + V_{step}$	0.1	81.3	95.8
	1	**82.9**	**96.8**

Table 4: Zero-shot results with different levels of guidance from verifiers. α comes from Eq. (5).

4.3 Ablation Study

4.3.1 Is guidance important during path searching reasoning?

We argued that it is important to introduce guidance in the loop during reasoning path searching. To validate this argument, we adjust the weight of reward provided by verifiers during reasoning. The experiments are conducted using models without self-thinking. Table 4 summarizes the performance on zero-shot datasets with different settings of guidance. For "w/o verifiers", the solutions are predicted by a generator only and applied with "Self-Consistency". As demonstrated in Table 4, guidance from V_{path} can provide performance gains on SingleOp, with a 20.6% absolute improvement. We further incorporate the guidance from the step-level verifier V_{step}. As described in Eq. (5), increasing the weight of reward (α) from V_{step}, CoRe achieves a higher accuracy on both SingleOp and MultiArith. Thanks to the feedback and guidance during the reasoning stage, the generator tends to explore more often on a path with a higher reward score. As a result, CoRe increases the accuracy on SingleOP from 59.6% to 82.9% and MultiArith from 92.3% to 96.8%.

4.3.2 How much does self-thinking boost the reasoning ability of a language model?

To examine the effect of self-thinking, we explore it along with two axes: 1) the number of iterations and 2) the type of search strategy. Since we
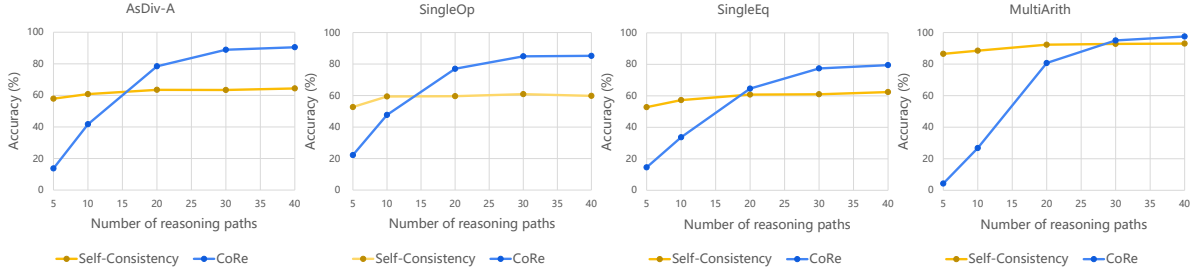
passes previous fine-tuned SoTAs on all datasets in a zero-shot setting. The results clearly demonstrate the capability of CoRe to greatly boost PLMs' reasoning ability.

4.2.3 GSM8K Results

Beyond improvements on zero-shot results, we observe that the fine-tuning setting can benefit a lot from our CoRe framework, as shown in Table 3. Compared to previous fine-tuned SoTA (Cobbe et al., 2021) (GPT-3 350B), CoRe outperforms it with much fewer parameters, computation and inference time. Note that it samples 100 solutions for each question while we only search 40 paths.

For a comprehensive comparison, we include few-shot results with large-scale PLMs due to a limited number of "fine-tune" competitors. With regard to few-shot methods applied on large-scale PLMs ($\geq$ 100B parameters), CoRe only underperforms PaLM-540B strengthened by chain of thought prompt and self-consistency, further proving the effectiveness of our method.

Figure 3: Zero-shot results with different search strategies in cooperative inference.

# of iterations	0	1	2
Generator only (Greedy)	29.9	34.7	**34.9**
Generator + Self-Consistency	42.0	43.1	**45.9**
CoRe	60.0	**63.2**	61.6

Table 5: Results on GSM8K with models undergone a different number of self-thinking iterations. Outcomes of various search strategies are provided.

# of iterations		SingleOp	MultiArith
Generator	Verifiers		
0	0	82.9	96.8
0	1	**85.2**	**97.5**
1	0	81.9	96.3
1	1	83.3	97.2

Table 6: Zero-shot results with a different number of self-thinking iterations for generator and verifiers respectively.

apply the self-thinking procedure on the GSM8K dataset, we investigate the performance of models under different settings on GSM8K, as shown in Table 5. First, increasing the number of iterations can always improve the performance for both greedy decode and self-consistency. Our CoRe reaches saturation in one round, which might be attributed to the fact that System 1&2 learns better and faster on self-generated data by collaborative working. Second, regardless of the search strategy, self-thinking consistently boost the model's performance, which verifies that self-thinking boost language model's reasoning ability.

4.3.3 Do self-thinking generalize to other datasets?

We have performed self-thinking on GSM8K and proved that it improves the model's reasoning ability in 4.3.2. Furthermore, we explore whether the improvement on GSM8K comes at the cost of performance degradation on other datasets, i.e. the model overfits the dataset. As presented in Table 6, we vary the number of self-thinking iterations for the generator and verifiers respectively and provide results on SingleOp and MultiArith. The results show that the performance of the generator suffers a little, but verifiers can eliminate this undesirable effect and benefit a lot from self-thinking. The best results are obtained when only the verifiers are further fine-tuned, with the 2.3% and 0.7% absolute improvement on the two datasets respectively. This observation implies that we can

economize on training costs and time where target datasets are not included in self-thinking data.

4.3.4 How performance varies as the number of search iterations for different search strategies changes?

As shown in Fig. 3, accuracy on 4 datasets consistently increases along with the growth of search iterations for both search strategies. However, the scaling curves of self-consistency and CoRe are quite different. The performance gain quickly saturates with self-consistency. Sampling 40 paths can not further improve the accuracy, while the scaling curve of CoRe is much sharper. Due to the heuristic algorithm that requires the model to continue exploring on the previously generated paths, CoRe starts from a relatively lower level in the beginning, whereas the accuracy quickly improves as the number of search iterations increases. The result demonstrates the effectiveness of CoRe in searching reasoning paths, with a fast growth curve and a slow saturation rate.

4.4 Case studies

4.4.1 Improvements from CoRe

A typical exemplar from GSM8K is presented in Table 7. Greedy decode fails to find a reasonable path due to the limited exploration in the output space. In contrast, self-consistency samples multiple reasoning paths randomly, resulting in a richer candidate set. Although it finds some right solu-

Question	Sophia and Rose went together to the market to buy onions and potatoes. Rose bought 4 times the number of onions and potatoes Sophia bought. If Rose bought 12 onions and 4 potatoes, how many onions and potatoes in total did Sophia buy at the market?	Score
Greedy Decode	Rose bought 12 onions * 4 = 12 * 4 = 48 onions. Rose bought 4 potatoes * 4 = 4 * 4 = 16 potatoes. Sophia and Rose bought 48 onions + 16 potatoes = 64 onions and potatoes in total. [ANS] 64.	—
Most appropriate SC Path	If Rose bought 12 onions at the market, then Sophia bought 12 / 4 = 3 onions. If Rose bought 4 potatoes at the market, then Sophia bought 4 / 4 = 1 potato. Since Rose bought 12 onions and 4 potatoes, Sophia bought 3 + 1 = 4 onions and potatoes in total. [ANS] 4.	—
Most consistent SC Path	Rose bought 4 x 48 = 192 potatoes. In total, Sophia bought 192 + 48 = 240 onions and potatoes at the market. [ANS] 240.	—
CoRe Path 1	Since Rose bought 12 onions, this means there were 12 / 4 = 3 onions that Sophia bought. The number of potatoes that Sophia bought is 4, meaning there were 4 / 4 = 1 potato that Sophia bought. The total number of onions and potatoes that Sophia bought is 3 + 1 = 4. [ANS] 4.	0.994
CoRe Path 2	The number of onions Rose bought is 12 onions / 4 = 3. Sophia bought 4 / 12 = 1 / 3 of the number of onions Rose bought. The number of potatoes Rose bought is 4 potatoes / 4 = 1.33 of the number of potatoes Rose bought is 4 potatoes / 12 onions = 0.33. The total number of onions and potatoes Rose bought at the market is 3 onions + 1 potatoes = 4. [ANS] 4.	0.363
CoRe Path 3	Let x be the number of onions Sophia bought. Rose bought 4 * x onions. The total number of onions and potatoes Rose bought is 4 * x + x = 12. 5 * x = 12. x = 4. Sophia bought 4 onions. [ANS] 4.	0.023

Table 7: GSM8K samples generated through different searching methods. SC refers to Self-Consistency. The erroneous parts of each path are highlighted in red and the corresponding correct parts within other paths are blue.

Question	Gunter is trying to count the jelly beans in a jar. He asks his friends how many they think are in the jar. One says 80. Another says 20 more than half the first one. A third says 25% more than the first one. What is their average guess?
No Self-Thinking	Half the first guess is 80 / 2 = 40. The third guess is 40 * .25 = 10. Their total guesses are 80 + 20 + 40 + 10 = 150. Their average guess is 150 / 4 = 37.5. [ANS] 37.5.
With Self-Thinking	Half the first friend's guess is 80 / 2 = 40. The second friend's guess is 40 + 20 = 60. The third friend's guess is 80 * 1.25 = 100. Their total guesses are 80 + 60 + 100 = 240. Their average guess is 240 / 3 = 80. [ANS] 80.

Table 8: An example of GSM8K, model with self-thinking reasoned correctly, while the non-self-thinking model generated a wrong reasoning path and therefore failed.

tions occasionally, without any guidance, it fails to explore more frequently on the high-quality paths, thus ending up with a wrong answer obtained by majority voting as shown in the fourth row.

As a comparison, results generated by CoRe are listed with their scores. Similar to random sampling, the reasoning paths might be partially illogical, even though the final answers happen to be correct. Despite this challenge, CoRe is capable of distinguishing those poor-quality paths from the superior ones thanks to the verifiers. Adhering to the philosophy of cooperative reasoning we have emphasized, the verifiers managed to harness the generator throughout the reasoning procedure with the help of MCTS. Therefore, CoRe enjoys not only the advantage of having a diverse candidate set, but also the merit of being wiser and efficient during reasoning path searching.

4.4.2 Improvements from Self-Thinking

Table 8 shows an example that the vanilla model failed to solve the given question, whereas after the self-thinking, the model rectified the faulty parts and successfully addressed it. This displays that self-thinking boosts language models' inner reasoning ability regardless of the search strategy, which is also proved in Sec. 4.3.2.

5 Discussion

Although we only fine-tune the language model on GSM8K due to the scarcity of QA datasets annotated with intermediate rationales, zero-shot results on several arithmetic datasets prove that basic reasoning capability is transferable across datasets within the same domain. This observation implies that when it comes to a new domain, we only need to collect a limited number of question-answer pairs with reasoning paths, model's reasoning ability can generalize to other unseen datasets and can be further strengthened by our approach CoRe according to the experimental results.

6 Conclusions

In this work, we mimic the dual system of human cognition to develop an effective reasoning framework for solving the math word problems. The proposed approach is consisting of two ingredients: the generator as System 1 and the verifiers as System 2, and overall reasoning is conducted based on their mutual reinforcement. From the robustness and generalization aspects, CoRe activates superior reasoning ability of LMs, and thus outperforms PLMs that are dozens of times larger.

Limitations

The outcome on multiple datasets verifies the powerful reasoning ability, which even works on models with only several billion parameters. However, our self-thinking procedure utilizes only one dataset, GSM8K, and the available training set size is only 7.5K. The main reason is the scarcity of high-quality datasets with rich reasoning paths. And, collecting such data incurs huge computation costs and expensive human resources. Another limitation is that we have not conducted experiments on bigger language models, such as GPT-3 and PaLM, due to the expensive usage costs and the fact of no open-source codes. In a nutshell, in the future, we will focus on collecting more high-quality labeled data and exploring our method on more powerful language models.

Ethics Statement

In this work, our CoRe shows impressive reasoning capability, however, it also comes with social risks. Here, we summarize three possible ethical impacts: i) PLMs with bias, ii) generated data with social stereotypes and iii) problematic data environments. Considering utilizing PLMs as backbones, several works present various potential risks in PLMs (Lucy and Bamman, 2021; Amin and Kabir, 2022). Fortunately, our method supports the replacement of different PLMs. Therefore, we encourage deploying some risk-free PLMs, expecting to reduce the potential ethical risks. Furthermore, once deploying harmful PLMs, the self-thinking process might generate several undesired data and those data are fed into language models, which deepens the bias and causes unintended social impacts. For reducing the aforementioned cases, we suggest recording generated sentences. In real-world applications, a good choice is to monitor generated content and then hand them over for human review. In addition to the two risks posed by PLMs, the data in downstream tasks is of great concern. In particular, private data might cause unpredictable influence because of their nature as a non-open source. Therefore, we believe that a data cleaning workflow is necessary to mitigate potential risks, such as PrivateClean (Krishnan et al., 2016). Finally, we encourage open debating about its utilization for increasing transparency and reducing the potential for misuse.

References

Akhter Al Amin and Kazi Sinthia Kabir. 2022. A disability lens towards biases in GPT-3 generated open-ended languages. *CoRR*, abs/2206.11993.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Johannes Czech, Patrick Korus, and Kristian Kersting. 2021. Improving alphazero using monte-carlo graph search. In *ICAPS*, pages 103–111. AAAI Press.

Jonathan St.B.T. Evans. 2003. In two minds: dual-process accounts of reasoning. *Trends in Cognitive Sciences*, 7(10):454–459.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *ICLR*. OpenReview.net.

Daniel Kahneman. 2011. *Thinking, fast and slow*. Farrar, Straus and Giroux.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *ECML*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *CoRR*, abs/2205.11916.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Sanjay Krishnan, Jiannan Wang, Michael J. Franklin, Ken Goldberg, and Tim Kraska. 2016. Privateclean: Data cleaning and differential privacy. In *SIGMOD Conference*, pages 937–951. ACM.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2022. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. In *AAAI*, pages 13188–13190. AAAI Press.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the advance of making language models better reasoners. *CoRR*, abs/2206.02336.

Chao-Chun Liang, Kuang-Yi Hsu, Chien-Tsung Huang, Chung-Min Li, Shen-Yu Miao, and Keh-Yih Su. 2016. A tag-based statistical english math word problem solver with understanding, reasoning and explanation. In *IJCAI*, pages 4254–4255. IJCAI/AAAI Press.

Li Lucy and David Bamman. 2021. Gender and representation bias in GPT-3 generated stories. In *Proceedings of the Third Workshop on Narrative Understanding*, pages 48–55, Virtual. Association for Computational Linguistics.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

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

Maxwell I. Nye, Michael Henry Tessler, Joshua B. Tenenbaum, and Brenden M. Lake. 2021. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In *NeurIPS*, pages 25192–25204.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752. The Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *AAAI*, pages 3082–3088. AAAI Press.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Trans. Assoc. Comput. Linguistics*, 3:1–13.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*. OpenReview.net.

Thomas Scialom, Paul-Alexis Dray, Jacopo Staiano, Sylvain Lamprier, and Benjamin Piwowarski. 2021. To beam or not to beam: That is a question of cooperation for language gans. In *NeurIPS*, pages 26585–26597.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen

Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. In *ICLR*. OpenReview.net.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models. *CoRR*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022c. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

A Dataset Details

The mathematical reasoning datasets with details are as follows (Detailed description of the statistics in Table 9). We follow the licenses for their papers.

The dataset in fine-tuning:

GSM8K (Cobbe et al., 2021) is a high-quality dataset with reasoning paths. It consists of 8.8K grade school math problems created by human writers, which are divided into a train set (7.5K) and a test set (1.3K). The reasoning paths include 2 to 8 steps with considering basic arithmetic operations. Furthermore, we conduct cooperative training and self-thinking on its training set.

The datasets in zero-shot inference:

ASDiv-A (Miao et al., 2020) includes diverse math word problems, which are required to answer a number for each question.

SingleOP (Roy et al., 2015) is proposed with elementary math problems of a single operation.

SingleEq (Koncel-Kedziorski et al., 2015) is construed with both single-step and multi-step math problems from mixed sources.

MultiArith (Roy and Roth, 2015) includes elementary math problems with multiple steps.

B Experimental Settings

B.1 Hyper-parameters Setting

For the generator and the step verifier, we train them for two epochs. The batch size is set to 16. The learning rate (LR) is set to $1e-5$ at the first epoch and $1e-6$ at the second epoch for generator. On the hand of step verifier we apply the warmup method then linearly decaying scheduler, LR is set to $1e-6$ and warmup ratio is 0.1.

For the path verifier, we train it for three epochs with batch size set to 128 and LR set to $1e-5$. Same LR scheduler as the step verifier has been applied for the path verifier. We set the gradient clip norm to 1.0 and the sampling temperature to 0.7. The random seed is set to 19990303 throughout the training process.

For MCTS, we set max search iterations to 40 during inference. In expansion, we search 20 tokens each time. In order to avoid expanding too many homogeneous children for the same node, we simply penalize the probability of first token if it has appeared in other child nodes. We set the max token number to 300 in roll out and limit the total token number of reasoning path to 400.

Dataset	# of samples	Avg # of words in questions
GSM8K	1319	46.9
ASDiv-A	1218	29.2
SingleOp	562	20.9
SingleEq	508	27.2
MultiArith	600	31.8

Table 9: Dataset statistics.

B.2 Details of Training Verifiers

Before two verifiers are fine-tuned, we utilize the generator to sample 100 solutions for each question following Cobbe et al. (2021). Then we train the two verifiers on the generated data as described in Sec. 3.2 Step 1.3.

B.3 Details of Self-Thinking

In each iteration of self-thinking, we initialize the model with the weights obtained from the previous round so as to save the computational costs. Since we use cooperative inference rather than random sampling to generate data for further training, solutions are expected more high-quality. Thus, the number of generated solutions M mentioned in Sec. 3.2 is set to 50 for saving computational cost and time. Due to the flexibility of MCTS, we have also tried to limit the time for searching rather than the number of iterations, which makes the total search time controllable and predictable. Moreover, this allows the model to adaptively adjust the final number of solutions searched for each question, due to the different levels of difficulty in questions. In our experiments, we observe that setting the time limit to 320 seconds provides better results than setting the iteration limit to 50, while maintaining approximately the same time consumption. Therefore, we use time control to generate data during self-thinking.

B.4 Baseline Settings

As shown in Table 1, the Instruct GPT-3 is based on text-davinci-002 version. Moreover, since Kojima et al. (2022) provides difference prompt setting, we list them in Table 10. For few-shot scenarios with the chain of thought prompts, we follow the original paper (Wei et al., 2022c).

C Extended Experiments

This section we replicate the work of Cobbe et al. (2021) with GPT-J and report the results in Table 11 for comprehensive comparison. CoRe fully surpasses Cobbe et al. (2021) when the number of

Backbone	Method	Reasoning Extraction Prompts	Answer Extraction Prompts
Instruct GPT-3 175B	zero-shot	Let's think step by step.	The answer (arabic numerals) is
	zero-shot*	Let's think step by step.	The answer is
	zero-shot-CoT	Let's think step by step.	The answer (arabic numerals) is
	zero-shot-CoT*	Let's think step by step.	The answer is
PaLM 540B	zero-shot	Let's think step by step.	The answer (arabic numerals) is
	zero-shot-CoT	Let's think step by step.	The answer (arabic numerals) is
	+ Self-Consistency	Let's think step by step.	The answer (arabic numerals) is

Table 10: Prompt setting for few-shot baselines.

# of reasoning paths	ASDiv-A	SingleOp	SingleEq	MultiArith
Cobbe et al.				
5	71.9	70.5	68.5	92.3
10	76.9	73.1	74.6	95.0
20	79.6	74.6	76.0	95.5
30	81.4	76.2	76.2	95.2
40	81.4	76.9	78.1	94.8
CoRe				
5	13.7	22.2	14.6	4.3
10	41.7	47.7	33.7	26.8
20	78.4	77.0	64.6	80.7
30	88.9	84.9	77.4	95.0
40	**90.5**	**85.2**	**79.5**	**97.5**

Table 11: Comparison between Cobbe et al. (2021) and CoRe with GPT-J as backbone model. The best scores are in **bold**.

reasoning paths reaches 30 and maintains a faster increasing rate after that. As a result, CoRe has a superior performance over Cobbe et al. (2021) on all the datasets and achieves a 9.1% and 8.3% improvement compared to it on ASDiv-A and SingleOp.

D   Future Work

We focus on measuring our method in boosting the language model's arithmetic reasoning ability in this work. Nevertheless, we believe that our framework can also be applied to other reasoning tasks seamlessly, e.g., commonsense reasoning and symbolic reasoning. We choose arithmetic reasoning because it is the fundamental type of reasoning task. Additionally, we believe solving arithmetic reasoning is the first step toward a general cognitive reasoning system. In the future, we will explore other reasoning tasks and put more effort into low-resource scenarios.

A For every submission:

☑ A1. Did you describe the limitations of your work?
After conclusion section

☑ A2. Did you discuss any potential risks of your work?
After limitations section

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

4.1

☑ B1. Did you cite the creators of artifacts you used?
4.1

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix A

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Ethics Statement

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Ethics Statement

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
4.4

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4.1

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4, Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4.1

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.