Hint-before-Solving Prompting: Guiding LLMs to Effectively Utilize Encoded Knowledge

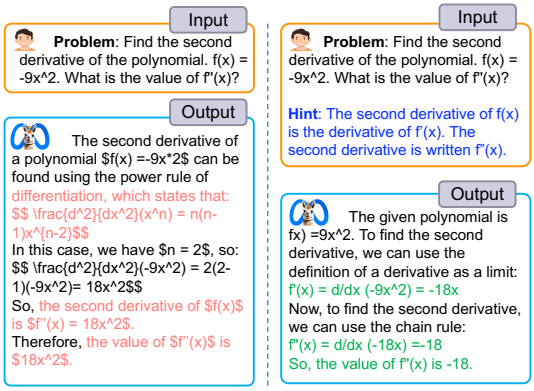
Anonymous ACL submission

Abstract

Large Language Models (LLMs) have recently showcased remarkable generalizability in various domains. However, they still encounter challenges in tasks that involve reasoning, especially those requiring multiple-step reasoning. Despite their extensive knowledge, LLMs still face challenges in efficiently utilizing encoded knowledge to develop accurate and logical reasoning processes. Additionally, there is currently limited research in this area. To mitigate this problem, we introduce Hint-before-Solving Prompting (HSP), a method that guides Large Language Models (LLMs) to first generate hints aiding in problem-solving, and then produce solutions that include intermediate reasoning steps. The hint can be specific knowledge, pivotal concepts, or analytical insights essential for solving the problem. Since HSP is orthogonal to prompting methods (e.g., Chain-of-Thought (CoT)), we applied HSP to CoT, Least-to-Most, Plan-and-Solve, and Standard promptings. We conducted extensive and comprehensive experiments on six reasoning benchmarks and four open-source models (Llama2 family models and Mixtral). The results show that HSP significantly improves the accuracy. The average across six datasets, Llama2-70B-Chat, improved by 1.5 under HSP-enhanced CoT, and the improvement jumped to 9.7% with high-quality hints. Beyond exploring training-free LLM capabilities, we built the HSPMATH dataset based on HSP and fine-tuned Llemma-7B, reaching 61.7% accuracy, surpassing GPT-3.5. We make our code and dataset publicly available at `https://github.com/sfhff216/hsp`

1 Introduction

Benefiting from extensive training corpora and computational resources, Large Language Models (LLMs) have reached state-of-the-art performance in numerous Natural Language Processing (NLP) tasks (Touvron et al., 2023a; OpenAI, 2023; Touvron et al., 2023b; Zhao et al., 2023b; Mistral AI Team, 2023). However, LLMs still face challenges in complex reasoning tasks, such as mathematical reasoning (Lu et al., 2023; Luo et al., 2023a; Imani et al., 2023) and commonsense reasoning (Paranjape et al., 2021; Sap et al., 2020). Although possessing a wealth of knowledge, LLMs always fail to accurately apply encoded knowledge to generate coherent and strongly logical reasoning chains when addressing reasoning tasks.



(a) Without Hint (CoT) (b) With Hint (CoT +Hint)

Figure 1: The output comparison of Llama-2-Chat-70B solving a math problem (calculus) with and without a hint. Red text indicates erroneous information; green text indicates correct reasoning. Findings: (1) having a hint can help the LLM understand the problem. (2) The LLM possesses knowledge of calculus, and with a hint, it can accurately apply this knowledge.

To improve the performance of LLMs on complex reasoning tasks, existing works have made several attempts. These previous works include fine-tuning on complete training datasets (Luo et al., 2023a; Yu et al., 2023; Yue et al., 2023) , training-free methods based on prompt engineering (Zhou et al., 2023a; Wang et al., 2023a; Fu et al., 2023; Lyu et al., 2023; Zhao et al., 2023a), or enhancing by retrieving knowledge from external knowledge bases (Yao et al., 2023b; He et al., 2023; Yang et al., 2023). Supervised fine-tuning methods are resource-intensive; current prompt engineer-

ing seldom attempt to improve LLMs' ability to use accurate knowledge; retrieval augmentation methods are limited to specific tasks. For example, mathematical reasoning that includes many special symbols is difficult to access relevant knowledge through keyword or semantic retrieval.

To mitigate these problems, in this work, we explore how LLMs can effectively utilize their encoded knowledge to enhance their reasoning logic and accuracy. We found that providing LLMs with hints effectively guides their use of encoded knowledge for problem-solving. Fig. 1 illustrates this by comparing Llama2-70B's outputs on a *calculus problem* with and without hints. The LLM cannot utilize *calculus knowledge* to solve the problem without any hints, as shown in Fig.1-(a). However, when given a hint (as shown in Fig.1-(b)): "... *The second derivative is written* $f''(x)$." the LLM can accurately apply its "*calculus knowledge*" to generate a correct and logical solution with intermediate reasoning. The reason can be attributed to that the hint suggested that "$f''(x)$ denotes the second derivative", which helped the LLM to better understand the target of the problem. Moreover, we conducted quantitative analysis on six reasoning datasets by introducing hints generated by GPT-4. The experimental results are shown in Fig. 3. We can find that giving high-quality hints can effectively improve reasoning performance.

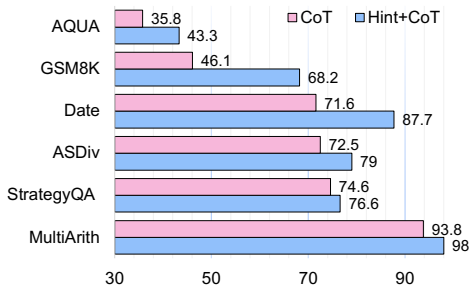


Figure 2: Results for Llama-2-Chat-70B (under CoT prompting) with or without introducing high-quality hints across six reasoning datasets. Findings: introducing hints lead to significant improvements, with an average relative increase of 9.7%.

However, it is challenging to provide high-quality hints for every sample. To address this problem, we propose the Hint-before-Solving (HSP) prompting method, which allows LLMs to generate hints on their own before solving a problem. The hints may include knowledge necessary for solving the problem (e.g., the hint shown in Fig. 1-(b)), analyzing the question, and providing essential ideas for the solution. Our explorations of Hint-before-Solving (HSP) Prompting in this paper are driven by following research questions:

Q1: *Can HSP guiding LLMs to autonomously generate helpful hints be effective?* To answer this question, we incorporated HSP into four well-performing prompting methods to investigate how HSP performs (EXP-I). Furthermore, we examined the effectiveness of the HSP variant, HSP2, which provides hints and solutions in two stages (EXP-II). And explore the upper bound of LLMs under the HSP2 framework (EXP-III). (Sec. 4.1)

Q2: *Does HSP still work when dealing with tasks that are challenging for LLMs?* In other words, if a task is difficult for LLMs, can they still provide helpful hints? To answer this question, we evaluated the challenging MATH dataset (EXP-IV). Furthermore, we explore how LLMs perform under the self-consistency setting (EXP-V). (Sec. 4.2)

Q3: *How do LLMs perform if they are supervised fine-tuned on a large-scale HSP prompting dataset?* To answer this question, we constructed the HSPMATH dataset based on GSM8K and conducted supervised fine-tuning on Llemma-7B and Llama-2-13B. The experimental results show that we achieved a performance of 61.7 on Llemma-7B, surpassing GPT3.5. (EXP-VI, Sec. 4.3)

The main contributions of this work are summarized as below:

(1) We discovered that providing hints allows LLMs to use their encoded knowledge accurately and effectively. For quantitative analysis, with GPT-4 generated hints, Llama-2-Chat-70B's accuracy increased by nearly 10% across six datasets.

(2) We propose the HSP prompting method, allowing LLMs to automatically generate useful hints. We conducted extensive experiments and analyses on applying HSP to four popular prompting methods to verify HSP's effectiveness.

(3) Beyond exploring the capabilities of training-free LLMs, we collected 75k samples with hints, namely HSPMATH, and conducted SFT experiments. Results show that HSP helps LLM learn faster.

2 Hint-before-Solving Prompting

The prominent Chain-of-Thought (CoT) (Wei et al., 2022) prompting method has inspired various prompting techniques to improve the LLMs' performance. Such as Least-to-Most (Zhou et al., 2022), tree-of-thought (Yao et al., 2023a), graph-of-thought (Besta et al., 2023), plan-and-solve prompt-

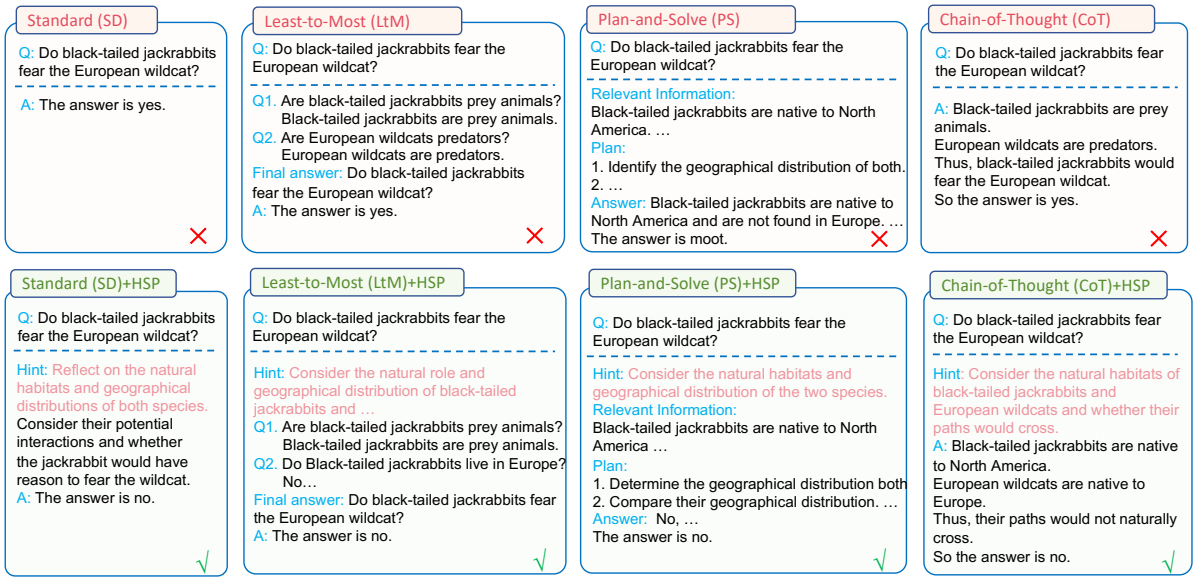


Figure 3: Examples of input and output before (four examples at the top) and after (four examples at the bottom) applying HSP to standard Least-to-Most, Plan-and-Solve, and CoT promptings. The red text in the textbox indicates hints. We find that hints from LLMs, including problem-solving ideas close to the correct answer (e.g., geographical distributions of both species), guide LLMs to use accurate knowledge for correct and logical reasoning.

ing (Wang et al., 2023a). In this work, we aim to design a new prompting method that allows LLMs to better utilize their encoded knowledge, namely Hint-before-Solving Prompting (HSP). HSP enables LLMs to explicitly generate hints for solving problems. The hints can be knowledge or key ideas for solving the problem or analyzing the question, etc., and developing an accurate and logical intermediate reasoning process before predicting the final answer.

HSP can be used in conjunction with some of the existing natural language forms of prompting methods (e.g., CoT). Fig. 2 shows examples of HSP integrated with four existing prompting methods, namely standard prompting, Least-to-Most prompting (Zhou et al., 2022), Plan-and-Solve prompting (Wang et al., 2023b), and Chain-of-Thought prompting (Wei et al., 2022). We can observe that the current hints provide LLMs with perspectives for thought (e.g., *consider the geographical distribution of black-tailed jackrabbits* ...), enhancing the effectiveness of prompting methods with the introduction of HSP.

3 Experiment

3.1 Large Language Model

To verify the performance of our proposed method, we consider Mixtral-8x7B-Instruct-v0.1 (*Mix-56B*) (Mistral AI Team, 2023) and Llama-2-Chat (Touvron et al., 2023c) family models, where Llama-2-Chat-7B (*Lm2-7B*), Llama-2-Chat-13B (*Lm2-13B*), Llama-2-Chat-70B (*Lm2-70B*) were studied. Note, the italicized text in parentheses represents the abbreviated names of the models.

Number	G8K	ASDiv	MArith	AQUA	MATH	SQA	Date
Samples	1,319	2,097	596	254	5,000	2,290	359
Examples	8	8	8	8	4	6	10

Table 1: The number of test samples and prompting examples across seven datasets.

3.2 Datasets

We evaluated the effectiveness of HSP across multiple datasets for mathematical and common sense reasoning tasks. Tab. 1 shows the number of test samples for these datasets and the number of samples for prompting in a few-shot setting.

Mathematical Reasoning We considered five popular mathematical reasoning datasets, namely *GSM8K (G8K)* (Cobbe et al., 2021), *MultiArith (MArith)* (Roy and Roth, 2016), *AQuA* (Ling et al., 2017), *ASDiv* (Miao et al., 2021), and *MATH* (Hendrycks et al., 2021a).

Commonsense Reasoning Two common sense reasoning datasets were also taken into account, which are *StrategyQA (SQA)* (Geva et al., 2021) and *Date Understanding (Date)* (Srivastava et al., 2022).

3.3 Baselines

The baseline Prompting methods considered in this work are listed below:

(1) *Standard Prompting (SD)* (Brown et al., 2020) generates the answer for the given question without intermediate steps. (2) *Chain-of-Thought Prompting (CoT)* (Wei et al., 2022) generate step-by-step solutions to a given problem. (3) *Least-to-Most Prompting (LtM)* (Zhou et al., 2022) involves decomposing a complex problem into simple subproblems. (4) *Plan-and-Solve Prompting (PS)* (Wang et al., 2023b) aims to handle the multi-step reasoning task by planning and solving each plan target.

To validate the effectiveness of the our HSP, we reimplemented some previous prompting methods. *To ensure a fair comparison, we did not deliberately reproduce results reported in previous papers but rather aimed to maintain consistency in the experimental setup. For different prompting methods, we kept using the same set of demonstration samples and modified their format according to the prompting method.* To demonstrate the usability of the results reimplemented in our work, we conducted a performance survey on existing baseline prompting with LLMs of comparable strength to those studied in this paper, with the results presented in Appendix C.

3.4 Experimental Settings

Demonstration examples Under any prompting method, one dataset is used with the number of demonstration examples in all the experiments discussed in this work. Specifically, as shown in Tab. 1, there are 8 demonstration examples each of GSM8K, ASDiv, MArith, and AQUA, 6 examples for StrategyQA, 10 examples for Date, 4 examples for MATH.

Hyperparameters of Greedy Decoding We use the vllm library [1] for few-shot evaluation. For greedy decoding, the hyperparameters are set as: top_p=1, max_tokens=500, temperature=0, and the number of reasoning path n=1. For self-consistency, the number of reasoning path n is set to 4, 16, 32, 64, 128, and temperature = 0.4. Other hyperparameters are set the same as the greedy decoding. All inference experiments are based on four A100 GPUs.

4 Experiments

4.1 Q1: Can HSP Work?

In this section, we considered three perspectives to answer whether HSP can enhance LLMs' performance by generating hints containing specific knowledge, pivotal concepts, or analytical insights critical for solving the problem before attempting to solve it. Next, we will illustrate the three perspectives in detail.

4.1.1 Exp-I: When HSP Meets Existing Prompting Methods

We applied HSP to four existing popular prompting methods to explore how HSP performs in different prompting methods. Our experimental prompting methods include standard prompting (SD), Least to Most prompting (LtM), Plan-and-Solve prompting (PS), and CoT prompting, as introduced in Sec. 3.3 The results are shown in Tab. 2. The main findings are summarized as below:

(1) *HSP is effective in standard and CoT prompting but fails in PS and LtM prompting.* From Tab. 2, we observe that the standard and CoT Prompting show significant performance improvements under HSP, while the enhancements from PS and LtM are limited. We try to give reasons below: Hints clarify the prompt or problem by offering key insights or solutions, influencing the logic behind the answers. They are crucial in task planning for both PS and LtM prompting, where introducing hints early can impact their planning process. Conversely, Standard and CoT prompting, focusing solely on the final answer or intermediate reasoning, are compatible with hints.

(2) *Larger model sizes tend to show more significant performance improvements.* From Tab. 3, we can observe that the average performance improvements for 7B, 13B, 56B, and 70B models across four prompting methods (e.g., CoT and LtM) are 0.5, 1.0, 1.3, and 1.5, respectively. The reason can be that the model capabilities increase as the size increases, and higher capabilities will help achieve higher quality hints for better problem-solving.

(3) *The introduction of HSP can steadily enhance the performance of CoT prompting.* We observe that CoT, combined with HSP, shows performance enhancements across all four models and six datasets, while SD, LtM, and PS all experience some scenarios of performance drop. From the line chart in Tab. 2, we can observe that LtM and PS

[1] https://github.com/vllm-project/vllm

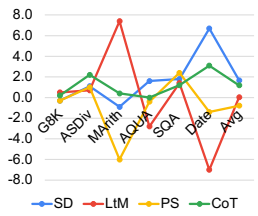
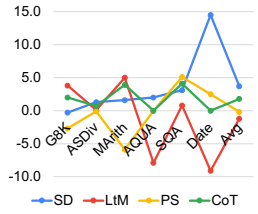
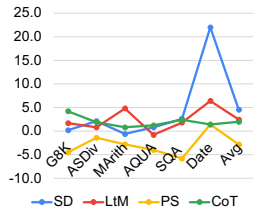
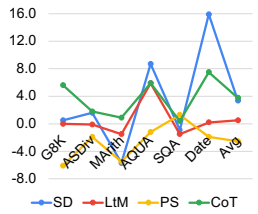
Method		HSP	G8K	ASDiv	MArith	AQUA	SQA	Date	Avg	Improvement
Lm2-7B	SD	×	5.8	43.7	7.4	19.7	62.0	33.1	28.6	
		✓ †	5.5	44.8	6.5	21.3	63.8	39.8	30.3	
	LtM	×	15.5	49.5	21.8	26.0	63.9	49.3	37.7	
		✓ †	16.0	50.2	29.2	23.2	65.3	42.3	37.7	
	PS	× ‡	21.8	55.8	66.6	25.6	58.1	34.8	43.8	
		✓	21.5	56.8	60.6	25.2	60.5	33.4	43.0	
	CoT	×	19.7	53.6	63.4	24.4	66.3	40.1	44.6	
		✓ †	19.9	55.8	63.8	24.4	67.5	43.2	45.8	
	Rlt Avg		0.0	1.2	0.2	-0.4	1.7	0.3	0.5	
Lm2-13B	SD	×	8.5	48.6	10.1	19.3	65.3	40.7	32.1	
		✓ †	8.2	49.9	11.7	21.3	68.4	55.2	35.8	
	LtM	×	23.8	55.8	52.7	31.1	68.8	60.4	48.8	
		✓ †	27.6	55.9	57.7	23.2	69.6	51.3	47.6	
	PS	× ‡	35.1	63.0	80.7	25.6	60.9	47.6	52.2	
		✓	32.4	62.9	74.8	25.6	66.0	50.1	52.0	
	CoT	×	34.5	60.5	83.2	25.6	68.0	57.7	54.9	
		✓ †	36.5	61.2	87.1	25.6	72.1	57.7	56.7	
	Rlt Avg		0.7	0.5	1.1	-1.5	3.3	2.0	1.0	
Lm2-70B	SD	×	12.6	60.6	26.3	24.8	72.9	54.6	42.0	
		✓ †	12.8	62.7	25.7	25.6	75.5	76.6	46.5	
	LtM	×	40.2	68.6	72.0	39.4	75.2	71.0	61.1	
		✓ †	41.9	69.4	76.8	38.6	77.0	77.4	63.5	
	PS	× ‡	60.0	74.1	95.8	40.2	64.7	62.4	66.2	
		✓	55.5	72.7	93.0	36.2	58.9	63.8	63.4	
	CoT	×	46.1	72.5	93.8	35.8	74.6	71.6	65.7	
		✓ †	50.3	74.4	94.6	37.0	77.0	73.0	67.7	
	Rlt Avg		0.4	0.9	0.6	-0.7	0.2	7.8	1.5	
Mix-56B	SD	×	19.8	64.3	44.6	22.0	72.1	45.4	44.7	
		✓ †	20.3	65.9	38.9	30.7	71.2	61.3	48.1	
	LtM	× ‡	56.0	77.1	74.3	43.3	73.9	64.1	64.8	
		✓	56.0	77.0	72.8	49.2	72.4	64.3	65.3	
	PS	× ‡	73.2	84.2	97.8	49.6	66.3	68.5	73.3	
		✓	67.1	82.3	92.3	48.4	67.6	66.6	70.7	
	CoT	×	63.7	78.3	96.1	42.5	74.7	69.9	70.9	
		✓ †	69.8	80.1	97.0	48.4	75.1	77.4	74.6	
	Rlt Avg		0.1	0.4	-2.9	4.8	-0.2	5.4	1.3	

Table 2: Results of applying HSP to existing prompting (Sec. 3.3). Green (pink) values indicate the best performance without HSP (with HSP). *Rlt Avg* denotes the average relative improvement on the four prompting methods. *Improvement* represents the relative performance improvement when introducing HSP compared to not using HSP. † indicates HSP significantly boosts performance, whereas ‡ suggests omitting HSP leads to better results.

exhibit significant fluctuations in average performance gains across each dataset, with numerous settings of negative improvement.

4.1.2 EXP-II: Effectiveness of HSP for CoT Prompting

In Exp-I, we found that applying HSP to CoT prompting results in significant and stable performance improvements across six datasets. Based on this, to identify flexible and effective ways to incorporate HSP, we attempted to explore whether a two-stage HSP (HSP2) approach could work in CoT prompting. The two-stage HSP means that LLMs produce outputs twice, first outputting a hint and then a solution. In contrast, HSP has only one output that contains both the hint and the solution. Experimental results on 6 datasets of 4 open source models are shown in Tab. 3. The main observations are summarized as below:

(1) *The performance of HSP and HSP2 is comparable, despite the different ways of introducing hints.* We can observe that among four LLMs, the largest average performance gap between HSP and HSP2 across six datasets was achieved on the Llama2-13B model with 0.5% (56.7-56.2). This indicates that although the methods of introducing hints differ, the extent of performance improvement brought by both is close.

(2) *HSP brings more stable improvements compared to HSP2.* From histograms in Tab. 3, HSP shows improvements on nearly every dataset under models of four different sizes. In contrast, HSP2 may lead to performance decreases in certain scenarios, for example, on the MArith dataset, the HSP2 performance decreases with Llama2-7B and Llama2-70B models.

Method		G8K	ASDiv	MArith	AQUA	SQA	Date	Avg
Lm2-7B	CoT	19.7	53.6	63.4	24.4	66.3	40.1	44.6
	+HSP†	19.9	**55.8**	**63.8**	24.4	**67.5**	**43.2**	**45.8**
	+HSP2‡	**22.6**	55.4	62.6	**25.2**	66.8	40.4	45.5
Lm2-13B	CoT	34.5	60.5	83.2	25.6	68.0	57.7	54.9
	+HSP†	**36.5**	61.2	**87.1**	25.6	**72.1**	**57.7**	**56.7**
	+HSP2‡	**36.5**	**61.9**	85.1	**26.0**	70.7	57.0	56.2
Lm2-70B	CoT	46.1	72.5	93.8	35.8	74.6	71.6	65.7
	+HSP†	50.3	**74.4**	**94.6**	37.0	**77.0**	73.0	67.7
	+HSP2‡	**54.3**	73.9	93.0	**37.8**	71.5	**76.0**	**67.8**
Mix-56B	CoT	63.7	78.3	96.1	42.5	74.7	69.9	70.9
	+HSP†	69.3	80.1	**97.0**	**48.4**	**75.1**	77.4	**74.6**
	+HSP2‡	**69.8**	**80.3**	96.8	45.7	74.3	**79.1**	74.3

Improvement

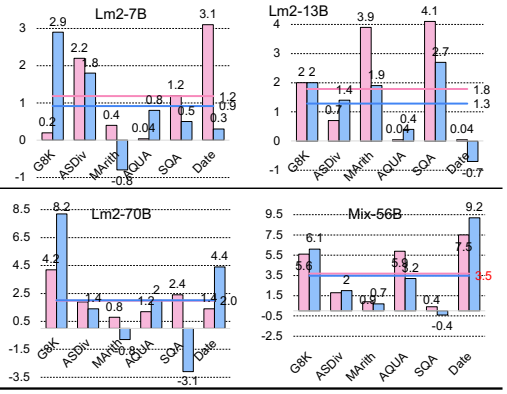


Table 3: The results of applying HSP and HSP2 in CoT prompting. The **bold** values indicate the best performance. † and ‡ denote that the performance of HSP and HSP2 is significantly better than CoT prompting, respectively.

Method		G8K	ASDiv	MArith	AQUA	SQA	Date	Avg
ChatGPT		79.1	-	97.3	55.1	-	-	-
7B	HSP2	22.6	55.4	62.6	25.2	66.8	40.4	45.5
	HSP2G	**39.0**	**62.5**	**88.9**	**28.7**	**69.5**	**61.0**	**58.3**
	Impv	16.4	7.1	26.3	3.5	2.7	20.6	12.8
13B	HSP2	36.5	61.9	85.1	26.0	70.7	57.0	56.2
	HSP2G	**56.4**	**66.4**	**95.6**	**36.6**	**72.0**	**69.4**	**66.1**
	Impv	19.9	4.5	10.5	10.6	1.3	12.4	9.9
70B	HSP2	54.3	73.9	93.0	37.8	71.5	76.0	67.8
	HSP2G	**68.2**	**79.0**	**98.0**	**43.3**	**76.6**	**87.7**	**75.5**
	Impv	13.9	5.1	5.0	5.5	5.1	11.7	7.7
56B	HSP2	69.8	80.3	96.8	45.7	74.3	79.1	74.3
	HSP2G	**79.5**	**84.1**	**99.2**	**56.3**	**76.5**	**84.7**	**80.1**
	Impv	9.7	3.8	2.4	10.6	2.2	5.6	5.7
Avg impv		15.0	5.1	11.1	7.6	2.8	12.6	9.0

Table 4: Experimental results of enhancing HSP2 with hints generated by GPT4. The values in green are the performance gap between HSP2G and HSP2. The blue values are the improvement across the four models. The values in bold represent the best performance.

4.1.3 Exp-III: The Impact of Hint Quality

Introducing HSP can effectively enhance the performance of CoT prompting. But what is the upper bound? Here, we choose to explore on HSP2 because it enables the hints from external sources, a feature not available in the one-stage HSP structure, and HSP2 is comparable in strength to HSP (Sec. 4.1.2). Hints generated by GPT-4 will be used as part of the input in the HSP2, denoting as HSP2G. Experimental results are shown in Tab. 4. The performance of ChatGPT is copied from Yin et al. (2023), where the number of examples used to evaluate GSM8K, MultiArith, and AQUA is 8, 8, and 4, respectively. The main findings are summarized as below:

(1) *High-quality hints make the open-source model outperforms ChatGPT.* We can observe that with the introduction of high-quality hints, all of the four LLMs with different model sizes and structures consistently showed performance improvement across six datasets. Furthermore, the Mix-56B equipped with HSP2(GPT4) outperformed ChatGPT on the GSM8K, MultiArith, and AQUA datasets.

(2) *The introduction of high-quality hints leads to more improvements in lower-capability models.* Tab. 4 shows that the average performance improvements for the Llama2 models sized 7B, 13B, and 70B are 12.8, 9.9, and 7.7, respectively. This indicates that with the support of high-quality hints, HSP2(GPT4)'s performance has improved a lot compared to HSP2. This can be attributed to that the low capability LLMs are hard to generate helpful hints that can assist in providing correct solutions. By providing high-quality hints, it is possible to offer more benefits beyond the capability of lower-ability LLMs. Therefore, there is a relatively large improvement in performance.

4.2 Q2: Can HSP work on Hard Tasks?

4.2.1 EXP-IV: Exploring Difficult Tasks

Prompting	Lm2-7B	Lm2-13B	Lm2-70B	Mix-56B
CoT	**4.5**	5.6	11.1	27.0
HSP	4.4	**5.7**	**11.4**	**28.6**†

Table 5: Results on MATH dataset. Values in bold denote the best performance, and the value with † denotes the performance of HSP significantly outperforms CoT.

The tasks we have explored are those that LLMs can handle well. As the difficulty of the task increases, LLMs may not possess sufficient knowl-

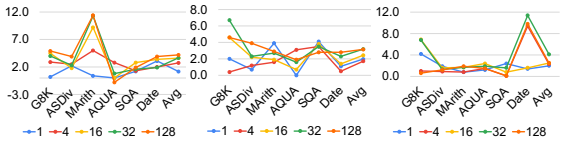
Param.	Prompting	Overall	Type							Level				
			AG	CP	GT	IA	NT	PG	PC	L1	L2	L3	L4	L5
n=1,t=0	CoT	27.0	39.01	18.99	18.58	13.4	16.85	47.07	15.57	62.47	44.41	30.59	18.62	8.08
	+HSP	28.6	39.09	23.21	21.09	13.84	15.93	46.27	15.2	64.3	45.64	30.33	18.29	8.91
	Impv	1.62	0.08	**4.22**	**2.51**	0.44	-0.92	-0.8	-0.37	**1.83**	**1.23**	-0.26	-0.33	0.83
n=4,t=0.4	CoT	31.9	46.67	26.58	22.55	15.39	20.56	52.47	17.95	71.17	49.33	36.6	23.39	10.8
	+HSP	33	47.35	26.37	26.1	15.39	21.3	54.88	19.6	72.31	51.45	36.34	25.86	11.33
	Impv	1.1	0.68	-0.21	3.55	0	0.74	**2.41**	**1.65**	**1.14**	**2.12**	-0.26	**2.47**	0.53
n=16,t=0.4	CoT	37.6	53.41	31.22	27.35	19.38	26.67	58.9	24.73	78.03	56.71	43.15	30.07	13.52
	+HSP	38.8	53.75	32.07	31.52	20.93	27.59	59.82	26.01	78.49	57.83	44.39	33.11	13.44
	Impv	1.2	0.34	0.85	**4.17**	**1.55**	0.92	0.92	**1.28**	0.46	**1.12**	**1.24**	**3.04**	-0.08

Table 6: The results of fine-grained evaluation for Mix-56B on the MATH dataset based on topic and problem difficulty. n is the number of sample paths of the self-consistency, and t is the temperature. *AG, CP, GT, IA, NT, PA, PC* respectively represent *Algebra, Counting & Probability, Geometry, Intermediate Algebra, Number Theory, Prealgebra, Precalculus*. Green values indicate an performance improvement of HSP prompting relative to CoT prompting, while red values indicate a decrease. Values in bold denote performance improvements greater than 1.

edge and capability to address it. This raises a research question: Can LLMs generate helpful hints when they meet the challenge task?

To answer this question, we chose to investigate the MATH dataset (Hendrycks et al., 2021b), which is a dataset that poses challenges for large language models. The results are shown in Tab. 5. We can observe that among the four models, only the Mix-56B model shows a significant improvement of 1.6 under HSP prompting compared to CoT prompting. However, the three models in the Llama-2 family model fail to improve the performance. The reason might be that the Llama-2 family models face significant challenges on the MATH dataset, with their best result being only 11.4 (Lm2-70B), while the Mix-56B model achieves 27.0 under CoT Prompting, it is difficult for Llama-2 family model to generate valuable hints.

To find which kind of samples Mix-56B can work, we performed a fine-grained analysis based on the mathematic problem topic and the difficulty, where the dataset provides the topics and the difficulty levels. Furthermore, to explore how self-consistency affects the performance, we evaluate this model using sample paths of n=4 and n=16 and a model temperature of 0.4. The results are shown in Tab. 6. The main findings can be summarized as: (1) As n increases, under the CoT+HSP setting, the samples for which the LLM sees performance improvements shift from low to high difficulty. (2) As n increases, it is commonly believed that the most challenging GT type experiences the most significant performance improvement, amounting to 4.17. These indicate that by increasing n, HSP enhancement will correctly solve more complex questions.



(a) 7B (b) 13B (c) 70B

Figure 4: The relative performance improvement of self-consistency between CoT+HSP and CoT. The numbers of sample paths are 4, 16, 32, and 128, and the model temperature is 0.4.

4.2.2 EXP-V: The impact of Self-consistency

In EXP-IV (Sec. 4.2.1), we found that self-consistency setting can improve performance of difficult tasks (MATH dataset), even difficult samples. This raises the question of how CoT prompting equipped with HSP performs under a self-consistency setting for the popular tasks. We sample paths with numbers (n) 4, 16, 32, and 128 for the self-consistency study and set the model temperature as 0.4. The relative improvement between CoT+HSP and CoT on six datasets is shown in Fig. 4 (Full results can be seen in the Appendix D). The main findings are as below:

(1) *As the number of sampling paths increases, the relative improvements brought by applying HSP also increase.* From Fig. 4, we can observe that at n=32 or n=128, all three models achieve their best performance. By calculating the Pearson correlation between the number of sampling (n) and relative performance for Lm2-7B, Lm2-13B, and Lm2-70B (excluding n=128), the correlations are 0.67, 0.72, and 0.95, respectively. The reason can be that the larger n leads to more explored hints, making it easier to generate hints beneficial for problem-solving.

(2) *Smaller models see the most significant relative performance improvement after applying self-consistency.* This might be because smaller models have lower capabilities, while with guided hints, increasing n makes it easier to correct originally incorrect solutions, thus leading to more substantial performance improvements.

4.3 Q3: How does SFT perform on HSP format datasets?

Despite the remarkable success of LLMs, most existing open-source LLMs (e.g., LLaMA-2) still face challenges in solving math problems due to complex reasoning processes. How do LLMs perform when they are supervised fine-tuning (SFT) on the HSP format dataset?

We construct a SFT dastaset. Specifically, we collected hints by GPT4 for the GSM8K training set with 7.5k samples. 75,000 samples that rewrite the original questions from the MetaMATH (Yu et al., 2023), are extracted. And the hints will be utilized to the derived questions. The dataset is named HSPMATH, and the original 7.5k samples will be used as standard samples, which we call HSPMATH-1.s The results with supervised fine-tuning on GSM8K under Llemma-7B and Llama2-13B are shown in Tab. 7. The baselines include: Llama2 (Touvron et al., 2023c), RFT (Yuan et al., 2023), Llemma (Azerbayev et al., 2023), WizardMath (Luo et al., 2023b), WizardLM (Xu et al., 2023), MetaMath (Yu et al., 2023), GPT-3.5 (OpenAI, 2023), PaLM (Chowdhery et al., 2023), Minerva (Lewkowycz et al., 2022), and Chinchilla (Hoffmann et al., 2022) We can observed:

(1) *Supervised fine-tuning on datasets with HSP allows LLMs to achieve significant performance improvements.* From Tab. 7, we can observe that in three groups of SFT (e.g., HSP-Llemma vs. Llemma on the HSPMATH1 dataset), the performance dramatically improves with HSP enhancement, which is 5.1, 12.3, and 3.0, respectively. The reason can be that supervised fine-tuning involving hints helps the model better utilize encoded knowledge during the reasoning stage, thereby improving the model's generalization ability.

(2) *Our best supervised fine-tuning results surpassed many popular LLMs, including GPT-3.5.* With supervised fine-tuning on the 75k HSP-formatted samples HSPMATH, we achieved a strong performance of 61.7, outperforming closed-source models such as GPT-3.5 (57.1), PaLM-540B (56.5), and Llama2-70B (56.8), as well as WizardMath-7B (54.9), which was fine-tuned on a large-scale mathematical corpus.

Model	Size	ACC	Model	Size	ACC
open source			*close source*		
Llama2	7B	14.6	GPT-3.5	-	57.1
Llama2	13B	28.7	PaLM	540B	56.5
Llemma	7B	36.4	Minerva	540B	58.8
Llama2	34B	42.2	Minerva	62B	52.4
RFT	7B	50.3	Chinchilla	70B	43.7
Llemma	34B	51.5	*HSPMATH-1 (7.5k samples)*		
RFT	13B	54.8	Llemma	7B	46.8
WizardMath	7B	54.9	HSP-Llemma	7B	51.9
WizardLM	13B	55.3	Llama2	13B	42.6
Llama2	70B	56.8	HSP-Llama2	13B	54.9
WizardMath	13B	63.9	*HSPMATH (75k samples)*		
MetaMath	7B	66.5	Llemma	7B	58.7
MetaMath	13B	72.3	HSP-Llemma	7B	**61.7**

Table 7: The results of supervised fine-tuning on GSM8K. The value in bold denotes best SFT result.

4.4 Related Work

Chain-of-thought (CoT) has given a lot of inspiration to many works and has made numerous attempts to explore high performance. These techniques include using programming languages to represent the reasoning process (Gao et al., 2023; Lyu et al., 2023), representing the reasoning process with complex structures such as trees or graphs (Yao et al., 2023a; Besta et al., 2023), task decomposition (Zhou et al., 2022; Khot et al., 2023) and combining different prompting (Liu et al., 2023; Zhou et al., 2023b).

Like our work, Zheng et al. (2023) proposed Progressive-Hint Prompting (PHP), which aims to enhance LLMs' effectiveness by introducing hints iterative, where the hint is a numerical value obtained from the previous solution (or base prompt's solution). Although our HSP guides LLMs to use encoded knowledge to provide some hints, our method is one stage and does not require multiple iterations.

Limitation

Here, we summarize some limitations of this paper, as follows: (1) The HSPMATH dataset was expanded by rewriting questions from GSM8K nine times, but our hints were generated based only on the original samples and applied to the nine rewritten samples. The rewritten samples might undergo

logical changes, making the introduction of hints less harmonious. There might be a risk of poor performance during supervised fine-tuning. In the future, we will refine this dataset carefully and release a new version. (2) Due to limitations in computational resources, this paper did not conduct supervised fine-tuning on models larger than 13B parameters in the SFT experiments, resulting in an incomplete exploration of HSP-enhanced supervised fine-tuning. We will undertake this exploration in the future.

References

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *CoRR*, abs/2310.10631.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models. *CoRR*, abs/2308.09687.

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

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. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv e-prints*, pages arXiv–2110.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Trans. Assoc. Comput. Linguistics*, 9:346–361.

Hangfeng He, Hongming Zhang, and Dan Roth. 2023. Rethinking with retrieval: Faithful large language model inference. *CoRR*, abs/2301.00303.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *CoRR*, abs/2203.15556.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. Mathprompter: Mathematical reasoning using large language models. In *Proceedings of the The 61st Annual Meeting of the Association for Computational Linguistics: Industry Track, ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 37–42. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh*

International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 158–167. Association for Computational Linguistics.

Tengxiao Liu, Qipeng Guo, Yuqing Yang, Xiangkun Hu, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2023. Plan, verify and switch: Integrated reasoning with diverse x-of-thoughts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2807–2822. Association for Computational Linguistics.

Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2023. A survey of deep learning for mathematical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14605–14631. Association for Computational Linguistics.

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

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

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *CoRR*, abs/2301.13379.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2021. A diverse corpus for evaluating and developing english math word problem solvers. *CoRR*, abs/2106.15772.

Mistral AI Team. 2023. Mixtral of experts. `https://mistral.ai/news/mixtral-of-experts/`. Accessed: 2023-12-26.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Bhargavi Paranjape, Julian Michael, Marjan Ghazvininejad, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Prompting contrastive explanations for commonsense reasoning tasks. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4179–4192. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *CoRR*, abs/1608.01413.

Maarten Sap, Vered Shwartz, Antoine Bosselut, Yejin Choi, and Dan Roth. 2020. Commonsense reasoning for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, ACL 2020, Online, July 5, 2020*, pages 27–33. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas

Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2609–2634. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *CoRR*, abs/2306.15626.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15135–15153. Association for Computational Linguistics.

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

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *CoRR*, abs/2309.05653.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023a. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5823–5840. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023b. A survey of large language models. *CoRR*, abs/2303.18223.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *CoRR*, abs/2304.09797.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023a. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jianpeng Zhou, Wanjun Zhong, Yanlin Wang, and Jiahai Wang. 2023b. Adaptive-solver framework for dynamic strategy selection in large language model reasoning. *CoRR*, abs/2310.01446.

A Ablation Study

A.1 Robustness analysis

Considering the impact that varying sets of examples may have on outcomes, the question arises: Is

Model	HSP	GSM8K				SQA			
		E1	E2	E3	E4	E1	E2	E3	E4
7B	×	20.2	15.2	18.0	17.0	61.2	56.6	63.9	60.9
	✓	**22.7**	**21.6**	**23.4**	**22.8**	**63.8**	**61.5**	**65.9**	**63.3**
13B	×	35.9	29.1	25.4	32.2	64.1	60.6	67.5	63.2
	✓	**37.1**	**34.7**	**35.1**	**36.5**	**67.4**	**62.0**	**68.2**	**65.9**
70B	×	53.7	54.1	54.4	54.0	71.1	65.1	75.1	68.2
	✓	**60.1**	**56.3**	**55.3**	**59.3**	**71.7**	**72.1**	**75.8**	**73.1**
56B	×	67.9	68.8	67.2	67.8	65.4	60.3	69.3	61.9
	✓	**69.1**	**69.1**	**68.2**	**68.8**	**67.3**	**64.5**	**70.6**	**66.8**

Table 8: Experimental results for CoT Prompting with and without HSP on the GSM8K and StrategyQA (SQA) datasets across various example groups (E1, E2, E3, and E4). Values in bold denote the best results. 7B, 13B, 70B, and 56B denote *Llama2-7B, Llama2-13B, Llama2-70B,* and *Mixtral-8*7B*, respectively.

the HSP framework effective with diverse example sets?

To investigate this, we conducted experiments on the GSM8K and StrategyQA datasets. We randomly chose four sets of examples from the testing set, with each set comprising 8 examples for GSM8K and 6 examples for StrategyQA. We then crafted hints and solutions featuring intermediate reasoning steps aided by GPT-4. These experiments were carried out on four LLMs: Llama2-7B, Llama2-13B, Llama2-70B, and Mixtral-8*7B. According to the results presented in Tab. 8, CoT+HSP consistently outperformed CoT across the GSM8K and StrategyQA datasets, with all four models showing significant performance enhancements across the four example sets. This demonstrates the robustness of the performance gains achieved by integrating CoT with HSP.

A.2 Length of Reasoning

HSP can improve the model's ability to solve problems. Can it make the model's solution output shorter? We compared the output solution lengths of CoT and HSP-CoT on 4 models in 6 data sets, including HSP and HSP2. Overall, among all 24 solution length comparison experiments, 18 experiments of HSP were shorter. Compared to CoT, HSP2 has 14 experiments that are shorter than CoT. As shown in Fig 5: (a) is the experimental comparison on llama2-7b-chat-hf, (b) is the experimental comparison on llama-13b-chat-hf, (c) is the experimental comparison on llama-70b-chat-hf, and (d) is the experimental comparison on mixtral-8x7b. This shows that HSP can improve model performance while making the solution length shorter. The possible reason is that HSP can enable model inference to reduce redundant or invalid output.

B Case Study

B.1 Prompt Example

Here are two examples of prompts we used in inference experiments based on few-shot evaluations in Tab 9. The first prompt example is from gsm8k experiment. The second prompt example is from strategyqa experiment.

B.2 Model Output Example

HSP can guide the model to generate thoughts on some key points of the problem, as well as common problem-solving formulas and problem-solving techniques, etc. so that the model can focus on the correct answer ideas and avoid falling into blind reasoning that ignores commonsense in the real world. There are seven examples to study the role of HSP in Tab 10.

Case1 is whether a Jujutsu expert can defeat a Janissary. CoT output tends to introduce the skills of Jujutsu and ignore the attributes of the Janissary. HSP clearly points out that combat should focus on historical background and weapons. Guide the model's problem-solving logic to accurate reasoning to generate correct answer.

Case2 indicates that the model answer needs to be correct based on the natural ecological cycle chain;

Case3 indicates that the model answer needs to be correct based on the objective world geographical conditions;

Case4 indicates that the model answer needs to be correct based on the mathematical perimeter calculation formula;

Case5 is the annual salary calculation problem. CoT's answer ignores the hourly wage data, so the problem-solving logic is completely wrong. HSP fully considers the valid data information and give accurate reasoning. It first calculates weekly salary, then annual salary, and guides the model to the correct answer.

Case6 indicates that the model answer needs to make a correct answer based on the knowledge points of geometric polygons;

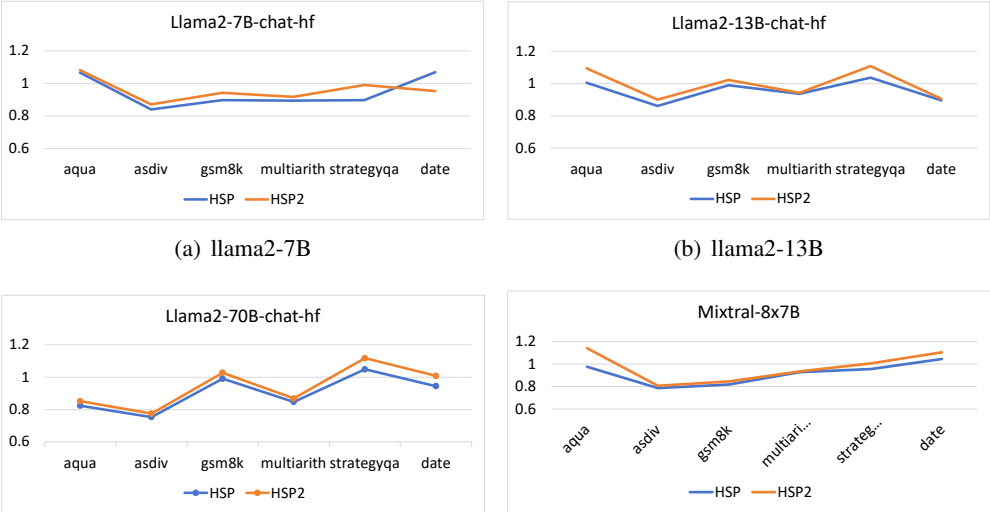
Case7 indicates that the model answer needs to focus on the calculation method of the ratio to make the correct answer;

Question	Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
Hint	Begin with the number of toys Shawn had initially. Then, add the number of toys he received from each parent. Remember, each parent gave him a certain number of toys, so you'll need to add those to his original amount to find out how many toys he has now.
Answer	Shawn started with 5 toys. If he got 2 toys each from his mom and dad, then that is 4 more toys. 5 + 4 = 9. The answer is 9.
Question	Do hamsters provide food for any animals?
Hint	Consider the natural role of hamsters in the food chain and who might rely on them as a source of nutrition.
Answer	Hamsters are prey animals. Prey are food for predators. Thus, hamsters provide food for some animals. \n So the answer is yes.

Table 9: Two prompt examples from gsm8k and strategyqa experiments

Question	Could a Jujutsu expert hypothetically defeat a Janissary?
True Answer	FALSE
CoT	Jujutsu is a martial art that focuses on using an opponent's strength against them.Janissaries were elite soldiers in the Ottoman Empire.Thus, a Jujutsu expert could potentially defeat a Janissary.\nSo the answer is yes.
CoT+HSP	Hint: Compare the martial arts skills of a Jujutsu expert with the combat training of a Janissary, considering the historical context and weaponry.\n\nA: A Jujutsu expert has martial arts skills, but no weapons.\nA Janissary is an elite Ottoman soldier with advanced combat training and weaponry.\nThus, a Janissary would likely defeat a Jujutsu expert.\nSo the answer is no.
Question	Does US brand Nice depend on Western honey bee?
True Answer	TRUE
CoT	A: Nice is a brand of ice cream.\nIce cream is not made from honey bees.\nThus, Nice does not depend on Western honey bees.\nSo the answer is no.
CoT+HSP	Hint: Consider the crops that Nice brand depends on and whether those crops require pollination by Western honey bees.\n\nA: Nice brand depends on almonds.\nAlmonds require pollination by Western honey bees.\nThus, Nice brand depends on Western honey bees.\nSo the answer is yes.
Question	Do black-tailed jackrabbits fear the European wildcat?
True Answer	FALSE
CoT	A: Black-tailed jackrabbits are prey animals.\nEuropean wildcats are predators.\nThus, black-tailed jackrabbits would fear the European wildcat.\nSo the answer is yes.
CoT+HSP	Hint: Consider the natural habitats of black-tailed jackrabbits and European wildcats and whether their paths would cross.\n\nA: Black-tailed jackrabbits are native to North America.\nEuropean wildcats are native to Europe.\nThus, their paths would not naturally cross.\nSo the answer is no.
Question	A point on the edge of a fan blade that is rotating in a plane 10 centimeters from the center of the fan. What is the distance traveled, in centimeters, by this point after 30 seconds when the fan runs at the rate of 300 revolutions per minutes?\n# Answer option: ['A)750pi', 'B)1500pi', 'C)1875pi', 'D)3000pi', 'E)7500pi']
True Answer	D
CoT	A: The fan makes 300 revolutions in 60 seconds. So it makes 300/60 = 5 revolutions in 1 second. So it makes 5*30 = 150 revolutions in 30 seconds. The point on the edge of the fan blade travels 10 centimeters in one revolution. So it travels 10*150 = 1500 centimeters in 30 seconds. So the answer is (b).
CoT+HSP	"Hint: The distance traveled by a point on the edge of a rotating object is calculated by multiplying the circumference of the circle it moves along by the number of rotations. A: The fan makes 300 revolutions in 60 seconds. In 30 seconds, it makes 300/2 = 150 revolutions. The circumference of the circle is $2\pi r = 2\pi(10) = 20\pi.Thedistancetraveledis150 * 20\pi = 3000\pi.Sotheanseris(d).$
Question	Jill gets paid $20 per hour to teach and $30 to be a cheerleading coach. If she works 50 weeks a year, 35 hours a week as a teacher and 15 hours a week as a coach, what's her annual salary?
True Answer	57500
CoT	A: Jill works 50 weeks a year.\nShe works 35 hours a week as a teacher, so that's 35 x 50 = $1750 per year.\nShe works 15 hours a week as a coach, so that's 15 x 50 = $750 per year.\nHer total annual salary is $1750 + $750 = $2500.\nThe answer is $2500.
CoT+HSP	Hint: To find out Jill's annual salary, first calculate how much she makes per week as a teacher and as a coach. Then, multiply this total amount by the number of weeks she works in a year. This will give you Jill's annual salary.\nA: Jill makes $20 per hour as a teacher, so 35 hours a week is 35 x 20 = 700 per week. As a coach, she makes $30 per hour, so 15 hours a week is 15 x 30 = 450 per week. 700 + 450 = 1150 per week. 1150 x 50 weeks = 57500 per year. The answer is 57500.
Question	James wants to hang crepe-paper streamers from his backyard fence for his birthday party. His backyard fence is a rectangle that measures 20 feet on the long side and 15 feet on the short side. How many feet of crepe paper does James need to buy?
True Answer	70
CoT	A: The fence is 20 feet long and 15 feet wide.\nThe total length of the fence is 20 + 15 = 35 feet.\nHe needs exactly the same amount of crepe paper as the length of the fence.\nSo he needs to buy 35 feet of crepe paper.\nThe answer is 35.
CoT+HSP	Hint: To find out how much crepe paper James needs to buy, calculate the perimeter of the fence. The perimeter is the total distance around the fence. To find the perimeter, add the lengths of all four sides of the fence together.\nA: The long side of the fence is 20 feet. The short side is 15 feet. So the perimeter is 20 + 15 + 20 + 15 = 70 feet. The answer is 70.
Question	The ratio of men to women on a bus is 5:9. If the total number of passengers on the bus is 84, and 20 women alight from the bus at the next stop, how many women have remained on the bus?
True Answer	34
CoT	A: In the beginning there were 84 passengers.\n20 women left, which means in total 20 + 84 = 104 men and women were on the bus before.\nThen we need to find a new ratio.\nLet's call the number of men m. We know that the number of women w must be 9/5 * m.\nWe know that the total number of passengers is 104, therefore m + w = 104.\nWe replace w in the equation, so m + 9/5 * m = 104.\nWe solve the equation and find m = 40.\nThat means there are 9/5 * 40 = 72 women on the bus at the beginning.\nIf 20 left, then 72 - 20 = 52 women stayed on the bus.\nThe answer is 52.
CoT+HSP	Hint: To find out how many women have remained on the bus, first determine the original number of women on the bus by using the given ratio. Then, subtract the number of women who alighted from the bus from this original number. This will give you the number of women who have remained on the bus.\nA: The ratio of men to women is 5:9. The total number of passengers is 84. So the number of women is 84 x 9/14 = 54. After 20 women alighted, 54 - 20 = 34 women remained. The answer is 34.

Table 10: Model output cases



(a) llama2-7B

(b) llama2-13B

(c) llama2-70B

(d) mixtral-8x7B

Figure 5: Comparison of solution length on six Datasets: HSP-CoT / CoT.

C Reference Baseline

In the experiment, we used four different prompt evaluation methods of SD, LtM, PS, and CoT as basleline, and generated a total of 96 evaluation scores on six datasets and four models. At the same time, in order to refer to previous scores using four different prompt methods on six datasets, we mainly collected acc scores published by some open source and closed source models.

The model parameters size used in our experiments ranges from 7B to 70B. The baseline scores of the four prompt methods for each dataset are introduced separately below.

On the aqua dataset: Our SD scores range from 19.3 to 24.8, referring to the 20.5 score of the UL2-20B model, and the highest score is consistent with the score of 24.8 of GPT-3 175B (text-davinci-002). Our LtM scores range from 26 to 43.3. The lowest score is 5.5 points higher than the 20.5 of the code-davinci-001 model, and the highest score is close to 44.5 of the text-davinci-003 model. Our PS scores range from 25.6 to 49.6, with the text-davinci-003 model's score of 42.5 falling in the middle of our scores. Our CoT scores ranged from 24.4 to 42.5, with the lowest score being higher than the 20.6 performance on the LaMDA 137B and the highest score being on par with code-davinci-002's 42.1 score.

On the asdiv dataset: Our SD scores range from 43.7 to 64.3. The lowest score is much higher than code-davinci-001's 38.5 and UL2 20B's 16, and the highest score is basically consistent with text-davinci-002's 64.8. Our LtM scores range from 49.5 to 68.6. The lowest score is nearly 3 points higher than the 46.6 of code-davinci-001, and the highest score is close to the 70 score of the text-davinci-003 model. Our CoT scores range from 53.6 to 78.3, with the lowest score exceeding code-davinci-001's 50.4 score by 3 points, and the highest score being close to code-davinci-002's 80 score.

On the gsm8k dataset: Our SD scores range from 5.8 to 19.8, with the lowest score being higher than code-davinci-001's 5.2, and the highest score exceeding code-davinci-002's 19.6 and Codex (text-davinci-002)'s 19.7. Our LtM scores range from 15.5 to 56, with the lowest score higher than code-davinci-001's 9.8, and the highest score far exceeding code-davinci-002's 38.3 and text-davinci-003's 34.6 scores by about 20 points. Our PS scores range from 21.8 to 73.2, including text-davinci-003's 58.2. Our CoT scores range from 19.7 to 63.7, higher than LlaMA2-7B's 14.6, LlaMA1-7B's 11, and LlaMA1-13B's 17.8, with the highest score close to WizardMath-13B's 63.9.

On the multiarith dataset: Our SD scores range from 7.4 to 44.6, with the lowest score close to code-davinci-001's 8.6 and the highest score exceeding code-davinci-002's 43.8 score. Our LtM scores range from 21.8 to 74.3, with the lowest score exceeding 18.5 for code-davinci-001 and the highest score exceeding 73.7 for text-davinci-003. Our PS scores range from 66.6 to 97.8, with the highest score exceeding text-davinci-003's score

of 87.2 by 10 points. Our CoT scores range from 63.4 to 96.1, with the lowest score exceeding code-davinci-001's 57.2, and the highest score being basically equivalent to code-davinci-002's 96.5.

On the strategyqa dataset: Our SD scores ranged from 62.0 to 72.9, with the lowest score exceeding the UL2 20B's 59. The highest score exceeded the PaLM 540B's 68.6. Our LtM scores range from 63.9 to 75.2, including 72.2 for code-davinci-002. Our PS scores range from 58.1 to 66.3, including 65.4 for text-davinci-003. Our CoT scores range from 66.3 to 74.7, with the lowest score exceeding 65.4 of LaMDA 137B and GPT-3 175B (text-davinci-002), and the highest score exceeding 73.2 of Codex (text-davinci-002).

On the date dataset: Our SD scores range from 33.1 to 54.6, with the lowest score exceeding LaMDA 137B's 21.5 and the highest score exceeding code-davinci-002's 51.3. Our LtM scores ranged from 49.3 to 71.0, with the highest score being close to code-davinci-002's 76.6. Our CoT scores ranged from 40.1 to 71.6, with the lowest score exceeding the LaMDA 137B's 26.8 and the highest score exceeding the PaLM 540B's 65.3

D Results of Self-consistency

Tab. 11 shows the results of self-consistency.

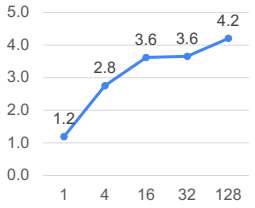
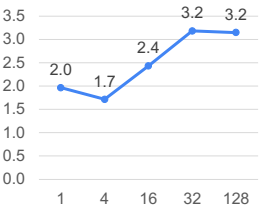
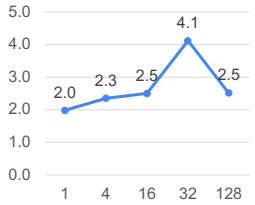
Model	SC	Hint	MATH				Commonsense		Avg	Relative Improvement
			GSM8K	ASDiv	MultiArith	AQUA	SQA	Date		
Llama2-7B	1	0	19.7	53.6	63.4	24.4	66.3	**40.1**	44.6	
	1	1	19.9	55.8	63.8	24.4	67.5	43.2	45.8	
	1	Impv	0.2	2.2	0.4	0.0	1.2	3.1	1.2	
	4	0	23.6	54.6	68.0	23.6	67.9	**40.1**	46.3	
	4	1	26.5	57.1	73.0	26.4	69.2	42.1	49.1	
	4	Impv	2.9	2.5	5.0	2.8	1.3	2.0	2.8	
	16	0	24.7	**55.5**	**68.5**	26.0	67.9	**40.1**	47.1	
	16	1	29.2	57.3	77.7	26.0	**70.7**	43.5	50.7	
	16	Impv	4.5	1.8	9.2	0.0	2.8	3.4	3.6	
	32	0	25.5	55.2	67.6	25.6	**68.6**	39.6	47.0	
	32	1	29.5	57.5	78.9	**26.4**	70.2	41.5	50.7	
	32	Impv	4.0	2.3	11.3	0.8	1.6	1.9	3.6	
	128	0	25.4	55.1	68.1	**26.4**	68.3	40.4	47.3	
	128	1	**30.3**	**59.0**	**79.5**	25.6	70.2	**44.3**	51.5	
	128	Impv	4.9	3.9	11.4	-0.8	1.9	3.9	4.2	
Llama2-13B	1	0	34.5	60.5	83.2	25.6	68.0	52.4	54.0	
	1	1	36.5	61.2	87.1	25.6	72.1	53.5	56.0	
	1	Impv	2.0	0.7	3.9	0.0	4.1	1.1	2.0	
	4	0	40.7	61.5	87.8	25.6	69.1	57.4	57.0	
	4	1	41.1	62.7	89.4	28.7	72.6	57.9	58.7	
	4	Impv	0.4	1.2	1.6	3.1	3.5	0.5	1.7	
	16	0	42.3	62.5	89.4	**28.0**	69.0	57.7	58.2	
	16	1	46.9	64.7	91.3	28.7	72.8	59.1	60.6	
	16	Impv	4.6	2.2	1.9	0.7	3.8	1.4	2.4	
	32	0	41.5	62.6	90.1	26.4	69.6	57.9	58.0	
	32	1	48.2	64.9	92.8	28.0	**73.1**	60.2	61.2	
	32	Impv	6.7	2.3	2.7	1.6	3.5	2.3	3.2	
	128	0	**47.9**	**62.9**	**90.1**	27.6	**70.2**	**58.8**	59.6	
	128	1	**52.5**	**66.8**	**93.0**	**29.5**	73.0	**61.6**	62.7	
	128	Impv	4.6	3.9	2.9	1.9	2.8	2.8	3.2	
Llama2-70B	1	0	46.1	72.5	93.8	35.8	74.6	71.6	65.7	
	1	1	50.3	74.4	94.6	37.0	77.0	73.0	67.7	
	1	Impv	4.2	1.9	0.8	1.2	2.4	1.4	2.0	
	4	0	59.5	75.0	95.3	39.8	78.2	73.3	70.2	
	4	1	60.5	75.9	96.1	41.3	78.3	82.7	72.5	
	4	Impv	1.0	0.9	0.8	1.5	0.1	9.4	2.3	
	16	0	60.1	76.3	96.1	42.1	78.4	73.3	71.1	
	16	1	67.0	77.9	97.8	44.5	79.2	74.9	73.6	
	16	Impv	6.9	1.6	1.7	2.4	0.8	1.6	2.5	
	32	0	60.6	77.1	96.3	45.3	78.5	72.7	71.8	
	32	1	67.4	78.4	98.0	**47.2**	**80.1**	**84.1**	75.9	
	32	Impv	6.8	1.3	1.7	1.9	1.6	11.4	4.1	
	128	0	**67.0**	**77.6**	**96.3**	**46.1**	**79.4**	**73.5**	73.3	
	128	1	**67.6**	**78.8**	**98.2**	47.6	79.5	83.3	75.8	
	128	Impv	0.6	1.2	1.9	1.5	0.1	9.8	2.5	

Table 11: The results of self-consistency on the six datasets. Values in green denote the relative performance improvement with hints versus without hints under the same setting. The blue bold values represent the best performance with hints, while the pink bold values indicate the best performance without hints. The figure on the right shows the average relative improvement across six datasets.