SELF-REASONING LANGUAGE MODELS: UNFOLD HIDDEN REASONING CHAINS WITH FEW REASONING CATALYST

Hongru Wang[1], **Deng Cai**[2], **Wanjun Zhong**[2], **Shijue Huang**[2],
Jeff Z. Pan[3], **Zeming Liu**[4*], **Kam-Fai Wong**[1*]
[1]The Chinese University of Hong Kong, [2]ByteDance,
[3]University of Edinburgh, [4]Beihang University
{hrwang, kfwong}@se.cuhk.edu.hk

ABSTRACT

Inference-time scaling has attracted much attention which significantly enhance the performance of Large Language Models (LLMs) in complex reasoning tasks by increasing the length of Chain-of-Thought. These longer intermediate reasoning rationales embody various meta-reasoning skills in human cognition such as reflection and decomposition, being difficult to create and acquire. In this work, we introduce *Self-Reasoning Language Model* (SRLM), where the model itself can synthesize longer CoT data and iteratively improve performance through self-training. By incorporating a few demonstration examples (i.e., 1,000 samples) on how to unfold hidden reasoning chains from existing responses, which act as a reasoning catalyst, we demonstrate that SRLM not only enhances the model's initial performance but also ensures more stable and consistent improvements in subsequent iterations. Our proposed SRLM achieves an average absolute improvement of more than $+2.5$ points across five reasoning tasks: MMLU, GSM8K, ARC-C, HellaSwag, and BBH on two backbone models. Moreover, it brings more improvements with more times of sampling during inference, such as absolute $+7.89$ average improvement with 64 sampling times, revealing the in-depth, diverse and creative reasoning paths in SRLM against the strong baseline.

1 INTRODUCTION

Recent studies have demonstrated that inference-time scaling (OpenAI, 2024) effectively improves performance of Large Language Models (LLMs) in various reasoning tasks (Wang et al., 2023a), such as mathematics and complex question answering, by extending the length of Chain-of-Thought (CoT) (Wei et al., 2022; Wang et al., 2023b) reasoning rationales (Qwen, 2024; DeepSeek-AI et al., 2025). With longer CoTs during inference, these LLMs could explore more creative and diverse reasoning rationales while assembling various meta-reasoning skills observed in human cognition (Gigerenzer, 1991; Johnson-Laird, 1988; Dye, 2011; Wang et al., 2024a), such as reflection and decomposition (Wu et al., 2024; Wang et al., 2025), resulting in better performance. However, the *scarcity* of these longer CoT data remains a significant obstacle to advancing inference-time scaling, especially for instructions without verifiable answers in the general instruction-tuning data.

Several prior studies have explored various approaches to get better CoT tuning datasets, where most of them utilize either the LLM itself (Wang et al., 2023d; Huang et al., 2023; Xue et al., 2025) or external reasoning models (Li et al., 2024a; Hu et al., 2024a) to generate (*or refine*) new (*or existing*) responses in the instruction-tuning dataset, leading to great improvements at various downstream tasks. Despite the effectiveness of these proposed methods, they still face several limitations. On the one hand, most of them focus on questions with verifiable answer such as math and code (Huang et al., 2023; DeepSeek-AI et al., 2025), being infeasible for general instruction-tuning dataset. On the other hand, another line of work typically assumes access to more powerful models to refine each sample iteratively (Xu et al., 2023; Li et al., 2024a). Such methods suffer from performance

*Corresponding Authors

plateaus or even degradation (Ding et al., 2024) and are inherently constrained by the capability ceiling of the powerful model.

To this end, we present *Self-Reasoning Language Models* (SRLM), which is capable to self-unfolding its own reasoning rationales and iteratively optimize itself, leading to enhanced overall capability. Specifically, we first create only few reasoning catalyst data that compose the demonstrations of how to enrich shorter CoT rationales into more longer and comprehensive CoT with the augmentation of various meta-reasoning skills. After incorporating the reasoning catalyst data with the original instruction-tuning data, the tuned model not only inherit the basic reasoning capabilities from the instruction-tuning dataset, but also learn how to refine reasoning simultaneously, resulting in *Self-Reasoning Language Models*. Consequently, the SRLM can refine its own reasoning rationales at each iteration with the processing of reasoning expansion and selection. During this process, the model generates enriched reasoning rationale candidates for the same instructions in the original instruction-tuning dataset. These rationale pairs are then filtered and selected using three proposed selectors without any prior assumption about the instruction and answer. Finally, the newly selected instruction-tuning dataset is combined with the reasoning catalyst data to create the training data for the next iteration of SRLM, which is initialized from the same base model. To conclude, our contributions can be summarized as follows:

- We introduce *Self-Reasoning Language Models* (SRLM), which is capable to act as both: 1) a response generation model (i.e., *learn to reason*); and 2) reasoning models to refine its own reasoning rationales (i.e., *how to reason*).
- We present *reasoning catalyst data*, function as demonstrations to guide the LLM to unfold all hidden reasoning chains with the augmentation of various meta-reasoning skills such as reflection and decomposition.
- Our experiments demonstrate that incorporating a small portion of reasoning catalyst data (less than 0.02%) enables small-size SRLMs to generate higher-quality instruction-tuning datasets compared to those generated by GPT-4o, while also achieving better performance and more stable improvements across iterations on various reasoning benchmarks.

2 Related Work

Synthetic Instruction-tuning Data. Previous research on instruction tuning has primarily utilized human-annotated data to create large, high-quality datasets (Wang et al., 2022; 2023b), being both time-consuming and labor-intensive. Therefore, recent studies turn to leverage more advanced models to automatically generate instruction-tuning datasets (Wang et al., 2023d; Peng et al., 2023; Li et al., 2024a; Wang et al., 2024b). For instance, Magpie (Xu et al., 2024) leverage open-source models to generate more diverse user queries and responses by only feeding the left-side templates up to the position reserved for user messages as input. Moreover, there is another line of works focused on refining existing instruction-tuning data using powerful models, such as WizardLM (Xu et al., 2023) and Reflection-tuning (Li et al., 2024a) and NILE (Hu et al., 2024a). In detail, Li et al. (2024a) utilize GPT-3.5 to reflect on existing instructions and responses while Hu et al. (2024a) use GPT4 to capture the knowledge gap between internal parametric knowledge in LLMs and external world knowledge in instruction-tuning data.

Self-Improved LLMs. Recent studies have demonstrated that LLMs can refine themselves using self-generated data, leading to enhanced performance and efficiency without human intervention, resulting in self-improving LLMs (Huang et al., 2023). It is crucial to select high-quality samples from the vast amounts of generated data. There are different ways to acquire the quality signals of data samples: 1) LLMs (Yuan et al., 2024); 2) ranking or reward models (Dong et al., 2023; Lu et al., 2024; Hu et al., 2024b); and 3) some automatic methods such as execution feedback (Haluptzok et al., 2022) or self-consistency in multiple reasoning paths (Wang et al., 2023c). One distinct aspect of our work is that we are working on general instruction-following data optimization iteratively, where the answers are not verifiable like math and coding problems, making most of previous studies infeasible and unaffordable (Huang et al., 2023; Li et al., 2024a).

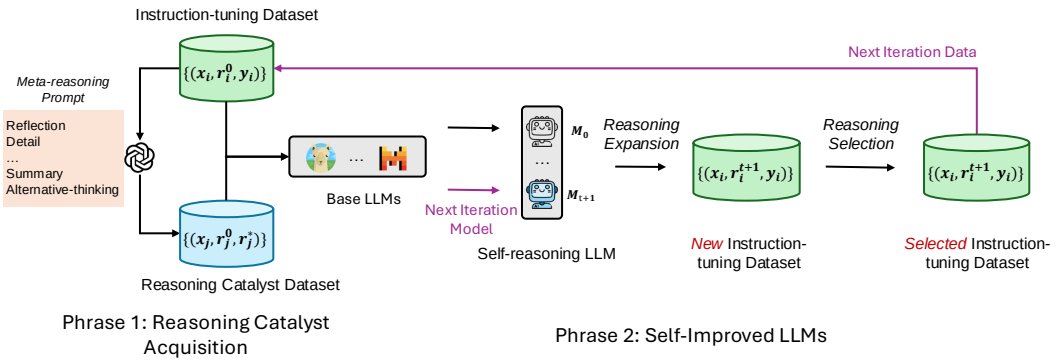

Figure 1: The framework of our proposed Self-Reasoning Language Models, which consists of two phrases.

3 METHOD

3.1 TASK FORMULATION

Given a large language model $\mathcal{M}$ and the original instruction-tuning dataset $D^0 = \{(x_i, r_i^0, y_i)\}$ at time step 0, where x_i is the instruction, r_i is the chain-of-thought rationale Wei et al. (2022) [1] and y_i represents the final answer in the i_{th} sample. The goal of self-improved LLMs is to iteratively refine the instruction-tuning dataset from the time step 0, leading to better dataset $D^t = \{(x_i, r_i^t, y_i)\}$ at time step t, ultimately resulting in enhanced performance at downstream tasks.

3.2 FRAMEWORK

As shown in Figure 1, there are two main phases in our method. In phase 1, we leverage existing LLMs to acquire seed data of reasoning augmentation, and function as reasoning catalyst to enlight the models to enrich existing reasoning rationales (\$3.2.1). In phase 2, the model then is fine-tuned using both original instruction-tuning data and reasoning catalyst data, and iteratively refined using self-generated new rationales, resulting in self-improved models (\$3.2.2).

3.2.1 REASONING CATALYST ACQUISITION

There is a Chinese proverb: "*It's better to teach a man to fish than to give him fish.*" Similarly, in the context of LLMs, rather than directly copying reasoning capabilities from existing instruction-tuning datasets, it is crucial to teach LLMs how to reason and refine their reasoning based on specific problems and existing reasoning rationales. To achieve this goal, we first carefully design meta-reasoning prompt [2], which is composed of different meta-reasoning skills such as *reflection*, *detail*, and *alternative-thinking*, inspired by meta-reasoning theory in cognitive science Zhou et al. (2024). Then we leverage these meta-reasoning skills to unfold hidden reasoning chains based on given instruction and previously reasoning rationales r_j^0, resulting in enriched reasoning rationales r_j^* as follows:

$$r_j^* = \mathcal{M}_{meta}(x_j, r_j^0) \tag{1}$$

Here the $\mathcal{M}_{meta}$ could be human experts or any off-the-shelf models. Therefore, it can effectively fill in missing details within existing rationales while preserving their core semantic meaning. Moreover, it is desirable to introduce new rationales (i.e., using *alternative-thinking*), correct errors in previous rationales (i.e., using *reflection*), and other hidden reasoning chains with different skills. Thus, this process provides exceptional flexibility and robustness to address errors, misleading reasoning paths, and unclear solutions in the previous reasoning rationale. Since we aim to teach the

[1] Note this is optional, take it as y_i if it does not exist.

[2] The prompt can be found in Table 6.

model how to reason instead of directly telling them the correct reasoning paths for all instructions. Therefore, we only incorporate a few samples (only 0.02% in the main experiment) as reasoning catalyst data to enforce the models' generalization to other unseen instructions.

3.2.2 SELF-IMPROVED LLMS

Given the original instruction-tuning dataset $D^t = \{(x_i, r_i^t, y_i)\}$ (t = 0 at the beginning) and fixed few reasoning catalyst data $R = \{(x_j, r_j^t, r_j^*)\}$, the original base LLM $\mathcal{M}$ [3] is then tuned using both types of data to lean reasoning and how to reason simultaneously, resulting in new LLM $\mathcal{M}_t$. Then we enrich the reasoning rationales in original instruction-tuning datasets (i.e., *iterative reasoning expansion*) and select better rationales to form the instruction-tuning dataset in the next step $t+1$ (i.e., *iterative reasoning selection*), which completes the whole iteration.

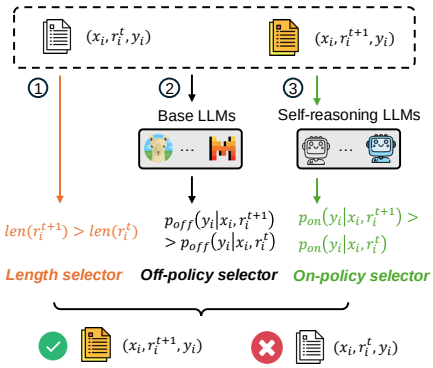

Figure 2: Three different types of reasoning selectors.

Iterative Reasoning Expansion. We directly prompt the M_t model to generate enriched reasoning rationales given previous reasoning r_j^t and x_j for *each sample* in the instruction-tuning dataset, distinguishing from reasoning catalyst acquisition.

$$r_{i,m}^{t+1} = \mathcal{M}_t(x_i, r_i^t), \quad \forall m \in \{1, 2, \ldots, N\}. \tag{2}$$

where N represents the number of sampling attempts, designed to enhance diversity and produce higher-quality output.

Iterative Reasoning Selection. It is important to determine which rationale is better for current instruction given the pair of (x_i, r_i^t, y_i) and (x_i, r_i^{t+1}, y_i). Thus we carefully design three different selectors to investigate corresponding effects as shown in Figure 2:

- **Length Selector.** We consider a longer rationale to be better because it provides more detailed insights into the previous reasoning. Even if it contains errors or irrelevant information, it can be refined and corrected in subsequent iterations.
- **Off-policy Selector.** We leverage the *original base model* to calculate the the probability of the answer y_i given the instruction x_i and different rationales, and then choose the rationale that has a higher chance of leading to the final answer.

$$r^* = \arg \max_{j \in \{t, t+1\}} P(y_i|x_i, r_i^j) \tag{3}$$

- **On-policy Selector.** Similar with off-policy selector, but we use the *Self-Reasoning Language Model* at last iteration to calculate the probability.

4 EXPERIMENTS

4.1 SET UP

Models. We mainly use two different base models: Llama3.1-8B (Grattafiori et al., 2024) and Mistral-7B-V0.3 (Jiang et al., 2023) since both of them are open-sourced and support longer context.

[3] We note we always finetune the model start from base models instead of continual tuning.

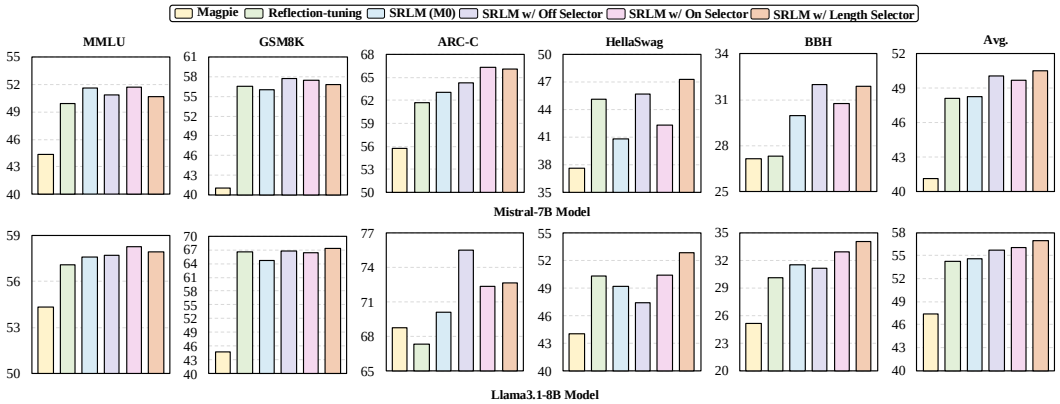

Figure 3: The performance of Self-Reasoning Language Models (SRLM) across five different benchmarks, along with the average performance (Avg.). We report the best performance during the five iterations for each selector, and the detailed experimental logs can be referred to at Table 7.

Data. There are two different parts of data used in our experiments: 1) *instruction-tuning data*: we first randomly sample 50k instances from Magpie-reasoning-150k (Xu et al., 2024) and then adopt reflection-tuning (Li et al., 2024a) to reflect its instruction and the responses using GPT-4o model; 2) *reasoning catalyst data*: we use the GPT-4o model to generate only additional 1k samples (only 0.02% of whole training dataset) to enrich the original rationales with various meta-reasoning skills. We compare the performance of various models fine-tuned using our method against two above instruction-tuning datasets 50k Magpie data and 50k Reflection-tuning data.

Evaluation and Metrics. We choose several existing benchmarks to validate the effectiveness and generalization of our proposed method. Specifically, we mainly focus on tasks necessniate complex reasoning such as general reasoning benchmarks: MMLU (Hendrycks et al., 2021), mathematical reasoning: GSM8K (Cobbe et al., 2021), commonsense reasoning: ARC-C (Clark et al., 2018), HellaSwag (Zellers et al., 2019) and other challenging tasks: BBH (Suzgun et al., 2022). We choose Accuracy as the evaluation metric and use the zero-shot prompting method to evaluate the performance of fine-tuned models, following lots of previous works (Grattafiori et al., 2024).

Implementation Details. We train 1,000 steps on 16 A100 80G GPUs for each method to ensure a fair comparison, considering the slightly different sizes of the dataset. During the training, we set the learning rate as 1e-05, batch size as 4, and gradient accumulation steps as 2 following Zheng et al. (2024). After obtaining the SRLM, we set the sample times N in Eq 2 as 5, and the maximum iteration times as 5 for all selectors. During the inference, we adopt the same setting with temperature as 0.2, top p as 0.9, and max tokens as 8192 to make a fair comparison.

4.2 Main Results

The main results are shown in Figure 3. Based on that, we have the following key observations.

Incorporating additional reasoning expansion samples into the instruction-tuning dataset acts as a *catalyst*, leading to improved performance. As we can find in the Avg. and most benchmarks, SRLM ($\mathcal{M}_0$) achieves better performance than Reflection-tuning and Magpie in both models. The only two exceptions are GSM8K and HellaSwag, due to the relatively smaller number of instances of mathematical and commonsense reasoning in the reasoning expansion data.

Small Self-Reasoning Language Models can generate high-quality rationales iteratively. It is evident that SRLM with different selectors at various iterations all leads to improvement compared with the initial SRLM ($\mathcal{M}_0$). Since the only difference between these SRLMs is the used instruction-tuning dataset, it strongly demonstrates that the SRLM with a small size (7B or 8B) can generate better instruction-tuning samples than GPT-4o (a.k.a., the instruction-tuning data at the 0_{th} step). It is encouraging to find that synthesized data by small language models still can provide stronger supervision signals than the latest powerful models. In addition, we find that Llama3.1-8B achieves

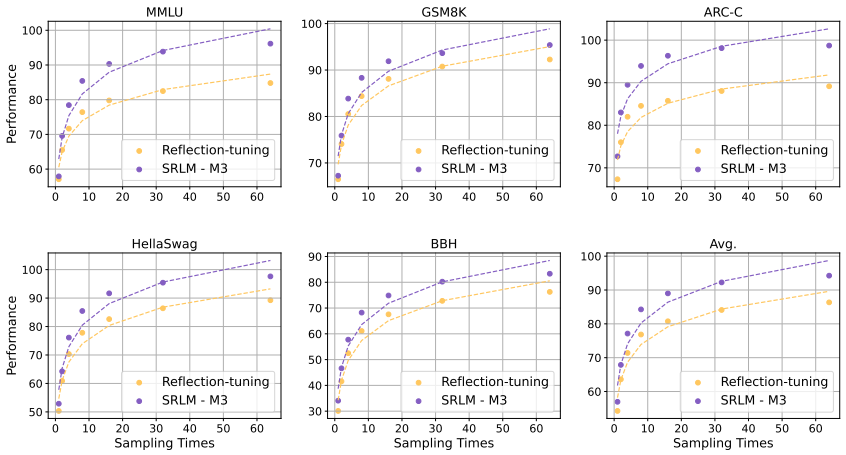

Figure 4: The performance of the baseline and our proposed SRLM with different sampling times ranging from 1 to 64 (i.e., 1, 2, 4, 8, 16, 32, 64) on five various benchmarks and the Avg. performance.

more consistent and higher improvement than Mistral-7B since it generates better samples due to more strong base capabilities.

All types of selectors outperform SRLM ($\mathcal{M}_0$), with the on-policy selector excelling in MMLU, while the length selector performs better on HellaSwag and BBH. Generally, Length selector achieves the best performance at Avg. for both models, revealing the excellence of this simple strategy in SRLM. In detail, we observe a seesaw phenomenon across different datasets and models. For example, with the same Mistral-7B model, the on-policy selector performs better on ARC-C but worse on HellaSwag. However, the on-policy selector on the Llama3.1-8B model performs worse compared to the other selectors on ARC-C. This can be attributed to the interaction effects between the models and datasets, highlighting the challenges in evaluating self-improving LLMs.

5 Analysis

5.1 Effects of Sampling Times.

In the main experiment, we mainly consider the best-of-1 performance of different methods. Alternatively, the best-of-N performance has been proved to be an effective way to showcase the depth and creativity of the model's problem-solving potential in terms of reasoning Chow et al. (2024). In this way, we conduct the multiple sampling ranging from 1 to 64, aiming to investigate the performance under the best-of-N setting. Figure 4 shows the performance of our proposed SRLM ($\mathcal{M}_3$ with length selector) and the baseline. The full experimental results can be found in Table 11 and fitting functions can be found in Appendix. First of all, it is worth noting that SRLM achieves much better performance with more sampling times on all benchmarks, as the gains get bigger and bigger. This is further supported by the larger coefficient of the logarithmic term in the fitted function of SRLM. Secondly, we can observe that MMLU achieves the highest growth rate and then followed by ARC-C and HellaSwag. As shown in Table 11, SRLM achieves 94.24% on the Avg. performance with 64 sampling times with MMLU (96.15% (+11.35)), ARC-C (98.72% (+9.56)), HellaSwag (97.62% (+8.38)). Based on above observations, it is believed that our proposed SRLM can explore more depth, diverse and creative reasoning paths toward the final correct solution, revealing its promising potential.

5.2 Effects of Different Number of Iterations

Figure 5 presents the Avg. Performance under different iterations with three types of selectors. There are several key findings. On the one hand, on-policy methods typically perform poorly at the beginning of the iteration process. This can be attributed to the limited number of samples - only

a few hundred - selected during the initial iteration. Moreover, as the process progresses from the first to the third iteration, the number of selected samples increases with the upward trend of the performance observed for both models in the figure. After the third iteration, performance diverges: the number of selected samples decreases for Mistral-7B, leading to a performance decline, while it remains constant for Llama3.1-8B, resulting in stable and saturated performance.

On the other hand, the length selector mostly leads to the best performance of the three selectors, and we also observe model degradation no matter which selector is chosen. After checking the number of selected numbers at each iteration, we find that the off-policy selector selects the most samples in the first iteration, nearly matching the original dataset size, but drops sharply to around 10k and continues decreasing. In contrast, the length-based selector also decreases, but at a slower and steadier rate. The last thing we want to emphasize is that it is evident that SRLM is capable of correcting previous rationales considering that there may be errors in the enriched rationales in previous steps, as we can find that length and off-policy selector achieve the best performance at 5_{th} iteration with the worst performance at 4_{th}.

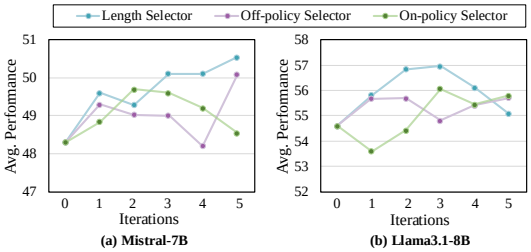

Figure 5: The effects of different number of iterations in Self-Reasoning Language Models.

5.3 Effects of Reasoning Catalyst Data

We also investigate several factors of reasoning catalyst data that may have serious effects on the final performance of SRLM, including the ablation study, different sizes, and sources of the data.

Ablation Study. We first directly remove the catalyst data from each iteration and train the model independently following the same setting in the main experiments. Figure 6 presents the final results at different iterations. Two obvious observations can be drawn from the results. First of all, the performance at later iterations mostly outperforms the performance at the initial iteration (iteration at $t = 0$) no matter incorporates a reasoning catalyst or not. This provides additional direct evidence that the newly generated instruction-tuning datasets at iterations ($t > 1$) are better than the GPT-4o generated dataset at iteration $t = 0$. Secondly, when incorporating reasoning catalyst data, the improvements become more stable and consistent. We can find that IT + Catalyst mostly achieves better performance than only instruction-tuning data, and it brings consistent improvement while the performance using instruction-tuning data only may fluctuate.

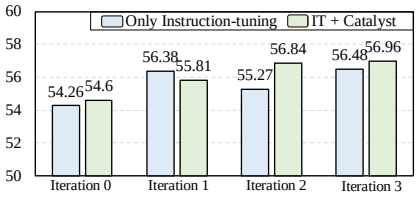

Figure 6: The ablation study by removing the reasoning catalyst data at each iteration. We report the Avg. performance here and the detailed results can be found in Table 8.

Size of Catalyst. We additionally random sample 5,000 and 10,000 from the original instruction-tuning dataset and build corresponding catalyst data. We also train the SRLM with 1,000 optimization steps to make a fair comparison following the same setting in the main experiments. Table 1 shows the final results. Generally, adding more reasoning catalyst data does not significantly improve average performance. However, it can lead to noticeable gains on specific datasets.

In detail, SRLM ($\mathcal{M}_0$) with 10,000 reasoning catalyst data achieves best performance at MMLU, GSM8K and BBH. Moreover, it continues to show superior results on MMLU with additional training iterations. This evidence suggests that incorporating more catalyst reasoning data benefits general reasoning tasks that require comprehensive evaluation, rather than domain-specific reasoning tasks. Furthermore, we can find that it offers limited benefits for commonsense reasoning tasks like ARC-C and HellaSwag, as the SRLM with just 1,000 data points consistently achieves peak performance across all iterations. We attribute this to relatively smaller ratio of commonsense reasoning samples in the sampled dataset.

Data	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
Reflection-tuning	57.08	66.49	67.32	50.34	30.09	54.26
SRLM – $\mathcal{M}_0$						
Cat. = 1k	57.60	64.59	**70.14**	**49.23**	31.46	**54.60**
Cat. = 5k	57.46	66.26	67.15	47.87	25.23	52.79
Cat. = 10k	**58.81**	**67.63**	67.41	46.30	**32.29**	54.49
SRLM – $\mathcal{M}_1$						
Cat. = 1k	58.31	65.73	72.87	**48.05**	**34.08**	55.81
Cat. = 5k	58.03	**67.85**	**74.23**	46.54	32.51	**55.83**
Cat. = 10k	**58.55**	67.63	73.29	45.27	31.78	55.30
SRLM – $\mathcal{M}_2$						
Cat. = 1k	58.27	**69.22**	72.70	**51.99**	32.03	**56.84**
Cat. = 5k	56.96	67.17	71.16	47.11	28.58	54.20
Cat. = 10k	**58.34**	67.40	**73.38**	51.06	**33.18**	56.67
SRLM – $\mathcal{M}_3$						
Cat. = 1k	57.91	**67.25**	**72.70**	**52.88**	**34.06**	**56.96**
Cat. = 5k	57.30	65.73	70.73	46.97	30.61	54.27
Cat. = 10k	**58.24**	67.17	70.56	50.87	32.83	55.93

Table 1: The results using different size of reasoning catalyst data based on Llama3.1-8B model with 1,000 steps optimization on three iterations ($\mathcal{M}_{0,1,2,3}$). We **bold** the best performance at each iteration.

Data	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
Reflection-tuning	57.08	66.49	67.32	50.34	30.09	54.26
IT_0 + New Cat	57.69	66.34	70.90	49.32	32.22	55.30
IT_1 + New Cat	56.77	68.31	69.28	45.53	33.40	54.66
IT_2 + New Cat	58.17	67.40	70.31	48.10	31.28	55.05
IT_3 + New Cat	56.45	66.49	69.62	46.47	30.27	53.86

Table 2: The performance of SRLM with length selector on Llama3.1-8B using *New* 1k reasoning catalyst data with 1,000 steps optimization.

New Catalyst. We also investigate the effects of applying out-of-distribution reasoning catalysts. To achieve this, we construct the catalyst using a separate randomly sampled 1,000 instruction-tuning dataset that has no overlap with the existing datasets. Table 2 shows the performance at each iteration. It is shown that incorporating additional out-of-distribution reasoning catalyst data also brings improvement compared with the only 4o-generated instruction-tuning data. However, consistent improvement in later iterations is not guaranteed as we observe only partial gains in certain benchmarks rather than a significant increase in overall average performance. We consider this due to difficulty in refining the rationale for existing instructions given out-of-distribution demonstrations.

Update Catalyst. Until now, we only update the instruction-tuning data set as shown in Figure 1. Alternatively, we can also update reasoning catalyst data by utilizing the newly generated rationales r_j^t. In detail, we can update the r_j^* only with r_j^{t+1} *or* update r_j^0 and r_j^* with r_j^t and r_j^{t+1} at the same time with simple length selector. However, we find that both of these two updating mechanisms can not bring significant improvement and even downgrade the performance at the later iterations as shown in Table 10. We consider the primary reason may be that LLMs tend to become constrained by their own outputs, limiting exploration of diverse solutions when more self-generated rationales are included. Moreover, the reasoning catalyst data introduced at the initial stage may be nearly optimal, thus being difficult to beat with only length selector.

6 Discussion

6.1 LLM-as-a-Judge Evaluation

Besides the exact match evaluation of five various reasoning benchmarks, we also conduct LLM-as-a-judge evaluation on two popular benchmarks: Arena-Hard Li et al. (2024b) and Alpaca Eval V2 Li et al. (2023). Following the default setting, we utilize gpt-4-1106-preview to compare the responses generated by different methods (i.e., Reflection-tuning and SRLM - $\mathcal{M}_3$ with length selector) against the response generated by the reference model (i.e., gpt-4-0314 in Arena-Hard, gpt-4-turbo in Alpaca Eval). Table 3 shows the win rate of response pairs. It is clear that our proposed SRLM achieves higher win rates at both datasets, leading to relatively 34% and 56% improvement on Arena-Hard and Alpaca Eval V2 respectively.

Method	Arena-Hard	Alpaca Eval V2
Alpaca-7B	-	2.59
Reflection-tuning	6.40	5.92
SRLM (*Ours*)	8.60	9.21

Table 3: The win rate of SRLM and Reflection-tuning

Data	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
Reflection-tuning	51.44	48.07	66.64	37.24	31.71	47.02
SRLM (*Ours*)						
IT_0 + Cat.	**55.34**	49.81	**69.28**	42.88	30.57	49.58
IT_1 + Cat.	53.85	**51.78**	67.58	44.93	**34.07**	**50.44**
IT_2 + Cat.	52.81	49.81	68.17	**47.35**	31.21	49.87
IT_3 + Cat.	52.31	47.08	66.47	45.03	28.53	47.89

Table 4: The results of Llama3.1-8B on instruction-tuning dataset (collected by length selector) of each iteration for Alpaca Taori et al. (2023) dataset with 2,000 optimization steps.

6.2 Effects of Instruction-tuning Data

To further validate the robustness of our proposed SRLM, we also conduct our experiments on another popular instruction-tuning dataset: Alpaca Taori et al. (2023). Following the same procedure, we finetuned our models for all datasets in 2,000 steps with batch size at 2 due to more longer responses in original Alpaca dataset. Table 4 shows the final results. It is clear that our method still outperforms the competitive 4o-generated data by reflection-tuning, and achieves best performance at the first iteration with the absolute $+3.42$ improvement on average performance of five different benchmarks. This result evidences the robustness and generalization of our proposed method on different datasets.

6.3 Inference Analysis

We also study the distribution of various meta-reasoning skills employed during training and inference stage of SRLM. Figure 7 shows the results. On the one hand, it is obvious that *detail* is the dominant reasoning skills used to unfold hidden reasoning rationales, and then followed by *decomposition*, *reflection*, no matter in training and inference stage. On the other hand, the striking similarity in reasoning skill distribution patterns between training and inference phases strongly suggests that instruction-tuning data construction critically influences model behavior. The distribution of various reasoning skills across different benchmarks during inferences also shows similar trends (as shown in Figure 8).

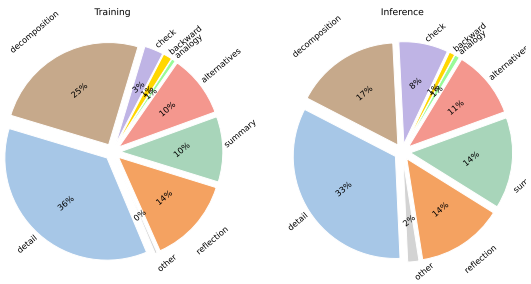

Figure 7: The distribution of various meta-reasoning skills employed at the training and inference of SRLM.

7 Conclusion

In this paper, we present Self-Reasoning Language Models (SRLM), which is capable to generate responses and unfold hidden reasoning chains at the same time, leading to self-evolved models that enhance performance across various reasoning benchmarks. By incorporating additional reasoning catalyst data with original instruction-tuning dataset to learn how to reason, SRLM can enrich previous reasoning rationales with the augmentation of many meta-reasoning skills in each iteration, i.e., *detail*, *decomposition* and *reflection*. Therefore, SRLM can iteratively refine its own rationales and evolves with better instruction-tuning dataset at each iteration. While there are many avenues left unexplored, we believe this is exciting and promising because SRLM is better able to self-discover diverse and creative reasoning paths, leading to more scalable ways to self-improve.

REFERENCES

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-n sampling in large language models, 2024. URL https://arxiv.org/abs/2412.15287.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yiwen Ding, Zhiheng Xi, Wei He, Zhuoyuan Li, Yitao Zhai, Xiaowei Shi, Xunliang Cai, Tao Gui, Qi Zhang, and Xuanjing Huang. Mitigating tail narrowing in llm self-improvement via socratic-guided sampling, 2024. URL https://arxiv.org/abs/2411.00750.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Vanessa Dye. Reflection, reflection, reflection. i'm thinking all the time, why do i need a theory or model of reflection?'. *Developing Reflective Practice: A guide for beginning teachers. Maidenhead: McGraw-Hill Education*, pp. 217–234, 2011.

Gerd Gigerenzer. From tools to theories: A heuristic of discovery in cognitive psychology. *Psychological review*, 98(2):254, 1991.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and Abhishek Kadian et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Patrick Haluptzok, Matthew Bowers, and Adam Tauman Kalai. Language models can teach themselves to program better. *arXiv preprint arXiv:2207.14502*, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Minda Hu, Qiyuan Zhang, Yufei Wang, Bowei He, Hongru Wang, Jingyan Zhou, Liangyou Li, Yasheng Wang, Chen Ma, and Irwin King. Nile: Internal consistency alignment in large language models, 2024a. URL https://arxiv.org/abs/2412.16686.

Minda Hu, Licheng Zong, Hongru Wang, Jingyan Zhou, Jingjing Li, Yichen Gao, Kam-Fai Wong, Yu Li, and Irwin King. SeRTS: Self-rewarding tree search for biomedical retrieval-augmented generation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1321–1335, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.71. URL https://aclanthology.org/2024.findings-emnlp.71/.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.67. URL https://aclanthology.org/2023.emnlp-main.67/.

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

Philip Nicholas Johnson-Laird. *The computer and the mind: An introduction to cognitive science*. Harvard University Press, 1988.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 16189–16211, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.958. URL https://aclanthology.org/2024.findings-acl.958/.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024b. URL https://arxiv.org/abs/2406.11939.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023.

Jianqiao Lu, Zhiyang Dou, WANG Hongru, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. Autopsv: Automated process-supervised verifier. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

OpenAI. Learning to reason with llms, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4, 2023. URL https://arxiv.org/abs/2304.03277.

Qwen. Qwq: Reflect deeply on the boundaries of the unknown, 2024. URL https://qwenlm.github.io/blog/qwq-32b-preview/. Accessed: 2025-02-01.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Hongru Wang, Lingzhi Wang, Yiming Du, Liang Chen, Jingyan Zhou, Yufei Wang, and Kam-Fai Wong. A survey of the evolution of language model-based dialogue systems, 2023a. URL https://arxiv.org/abs/2311.16789.

Hongru Wang, Rui Wang, Fei Mi, Yang Deng, Zezhong Wang, Bin Liang, Ruifeng Xu, and Kam-Fai Wong. Cue-CoT: Chain-of-thought prompting for responding to in-depth dialogue questions with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 12047–12064, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.806. URL https://aclanthology.org/2023.findings-emnlp.806/.

Hongru Wang, Huimin Wang, Lingzhi Wang, Minda Hu, Rui Wang, Boyang Xue, Yongfeng Huang, and Kam-Fai Wong. Tpe: Towards better compositional reasoning overnbsp;cognitive tools vianbsp;multi-persona collaboration. In *Natural Language Processing and Chinese Computing: 13th National CCF Conference, NLPCC 2024, Hangzhou, China, November 1–3, 2024, Proceedings, Part II*, pp. 281–294, Berlin, Heidelberg, 2024a. Springer-Verlag. ISBN 978-981-97-9433-1. doi: 10.1007/978-981-97-9434-8_22. URL https://doi.org/10.1007/978-981-97-9434-8_22.

Rui Wang, Fei Mi, Yi Chen, Boyang Xue, Hongru Wang, Qi Zhu, Kam-Fai Wong, and Ruifeng Xu. Role prompting guided domain adaptation with general capability preserve for large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2243–2255, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.145. URL https://aclanthology.org/2024.findings-naacl.145/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023c. URL https://arxiv.org/abs/2203.11171.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.340. URL https://aclanthology.org/2022.emnlp-main.340.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, 2023d.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Thoughts are

all over the place: On the underthinking of o1-like llms, 2025. URL https://arxiv.org/abs/2501.18585.

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

Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, Qunshu Lin, Junbo Zhao, Zhaoxiang Zhang, Wenhao Huang, Ge Zhang, Chenghua Lin, and J. H. Liu. A comparative study on reasoning patterns of openai's o1 model, 2024. URL https://arxiv.org/abs/2410.13639.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions, 2023. URL https://arxiv.org/abs/2304.12244.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing, 2024. URL https://arxiv.org/abs/2406.08464.

Boyang Xue, Qi Zhu, Hongru Wang, Rui Wang, Sheng Wang, Hongling Xu, Fei Mi, Yasheng Wang, Lifeng Shang, Qun Liu, and Kam-Fai Wong. Dast: Difficulty-aware self-training on large language models, 2025. URL https://arxiv.org/abs/2503.09029.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2024. URL https://arxiv.org/abs/2401.10020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 400–410, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed H Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. Self-discover: Large language models self-compose reasoning structures. *arXiv preprint arXiv:2402.03620*, 2024.

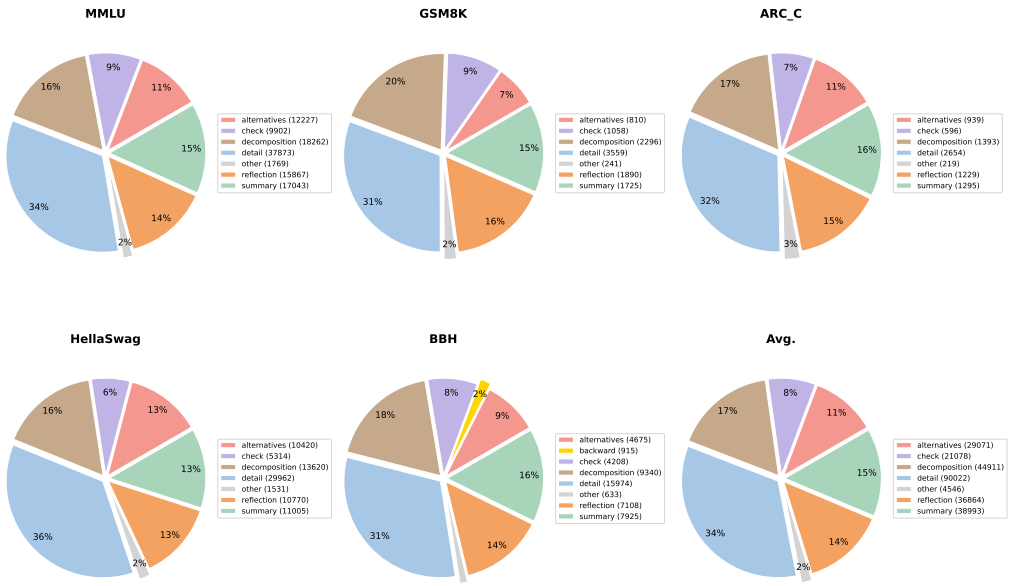

Figure 8: The distribution of meta-reasoning skills on five various benchmarks and the global distribution during inference.

A Appendix

A.1 Prompt Details

Table 5 and Table 6 show the instruction and system prompt respectively used to instruct LLMs to unfold the hidden reasoning chains with the augmentation of various meta-reasoning skills.

Here is the given question: {ins} Here is the original reasoning: {ans}

Table 5: The instruction used to prompt LLM to generate the enriched rationales.

A.2 Experimental Logs

Reflection-tuning. MMLU: $y = 6.4535 * log(x) + 60.5418$; GSM8K: $y = 6.0944 * log(x) + 69.6957$; ARC-C: $y = 4.8088 * log(x) + 71.8375$; HellaSwag: $y = 9.2765 * log(x) + 54.6557$; BBH: $y = 11.1345 * log(x) + 34.2507$; Avg.: $y = 7.5551 * log(x) + 58.1925$

SRLM. MMLU: $y = 9.0256 * log(x) + 62.8932$ GSM8K: $y = 6.5905 * log(x) + 71.4682$ ARC-C: $y = 5.9295 * log(x) + 78.0043$ HellaSwag: $y = 10.9284 * log(x) + 57.7607$ BBH: $y = 11.9599 * log(x) + 38.7086$ Avg.: $y = 8.8870 * log(x) + 61.7671$

A.3 Case Study

Table 12 shows the non-cherry pick test examples of MATH. It is obvious that our proposed SRLM can take advantages of different meta-reasoning skills such as *decomposition*, *detail*, *check*, *reflection* and *summary* to get the correct answer.

You are an expert at meta reasoning theory from cognitive science. Given the question, corresponding summarized reasoning and answer, you can always uncover hidden or unspoken reasoning, even when it isn't explicitly stated.
You need to add any missing reasoning thoughts that you think it is helpful or may occurs to understand and solve this question based on given summarized reasoning result. Your new reasoning should be more comprehensive, detailed and clear. Your answer should follow the format like <thoughts> your new reasoning here </thoughts>
Inside <thoughts> </thoughts>, you need to explicitly indicate which meta reasoning skill is used, some of them are shown below, but note you can use anything else if you think it is helpful or it naturally occurs when you solve this question.

<decomposition>: breaking down a complex problem into smaller, more manageable parts. Making sure that you also provide answers for all decomposed problems in this section. You can decompose iterativelly but should not contain same problem or exceed the max iteration depth which is three.

<backward>: starting with the desired observations at any previous reasoning step and working backward to identify the new reasoning directions.

<detail>: any details including but not limited to logic and reasons for your reasoning in this way, you are encouraged to add this at every unclear or unnatural reasoning step.

<summary>: summarize your reasoning to help future thinking.

<alternatives>: directly thinking in other ways, try to explore different solutions as much as possible to solve given problem.

<reflection>: you are encouraged to regularly reflect on your past reasoning in current response at various levels of detail, from sentence down to individual word. This will help you better understand and think through problems. It's okay to make mistakes; use them as opportunities to learn and improve.

<analogy>: you are encouraged to regularly consider other analogous problems with the problem you've encountered at various reasoning steps, along with their solutions. Reference existing theories or methods that guided your approach to solving these problems. These similar problems can be at various levels of detail - from larger overarching issues down to smaller sub-problems you encountered along the way. The key is to demonstrate a diverse range of problems and solutions, to show how you have approached and resolved challenges that are analogous to the current situation.

<check>: consider different edge cases or test cases carefully.

<other>: other meta reasoning skills you think is helpful or worthy to try to solve the task.

Notice:

1. All tags must be properly invoked and closed, using the format like <reflection> and </reflection>.

2. You should always use first-person perspective.

3. You can add any new meta reasoning skill at any positions except <reflection>. Note <reflection> can not be invoked without any reasoning in current response and it can be invoked at any positions when you already have some reasoning results.

4. You cannot change the original reasoning. However, if you identify any errors or improvements in the reasoning, you can add new reasoning steps using above meta-reasoning skills afterwards to correct or clarify the path, ensuring a better understanding and solution.

5. You can apply the same reasoning skills multiple times or use different skills simultaneously.

6. Your answer should start with <thoughts>, and end with </thoughts>.

Table 6: The system prompt used to initialize LLM to unfold all hidden reasoning chains.

Models	Methods	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
Mistral-7B	Magpie	44.35	41.02	55.72	37.61	27.13	41.17
	Reflection-tuning	49.85	56.56	61.69	45.15	27.29	48.11
	IT_0 + Cat. ($k = 0$)	51.63	56.03	63.05	40.82	29.97	48.30
	Length Selector						
	IT_1 + Cat. ($k = 1$)	49.50	58.91	62.37	45.91	31.33	49.60
	IT_2 + Cat. ($k = 2$)	50.01	59.59	64.85	42.16	29.82	49.29
	IT_3 + Cat. ($k = 3$)	50.56	56.41	64.59	46.88	32.11	50.11
	IT_4 + Cat. ($k = 4$)	49.07	58.45	60.41	42.03	29.35	47.86
	IT_5 + Cat. ($k = 5$)	50.66	56.79	66.13	47.30	31.87	50.55
	Off-policy Selector						
	IT_1 + Cat. ($k = 1$)	49.55	58.07	65.02	41.02	32.82	49.30
	IT_2 + Cat. ($k = 2$)	50.63	57.92	62.12	41.98	32.49	49.03
	IT_3 + Cat. ($k = 3$)	49.74	55.95	65.02	42.28	32.12	49.02
	IT_4 + Cat. ($k = 4$)	50.19	56.56	60.24	42.55	31.48	48.20
	IT_5 + Cat. ($k = 5$)	50.79	57.77	64.25	45.66	32.00	50.09
	On-policy Selector						
	IT_1 + Cat. ($k = 1$)	50.00	58.68	64.59	41.27	29.64	48.84
	IT_2 + Cat. ($k = 2$)	51.69	57.47	66.30	42.29	30.77	49.70
	IT_3 + Cat. ($k = 3$)	50.28	56.33	64.59	46.44	30.36	49.60
	IT_4 + Cat. ($k = 4$)	51.66	55.34	65.36	44.55	29.07	49.20
	IT_5 + Cat. ($k = 5$)	49.54	55.57	62.63	43.00	32.00	48.55
Llama3.1-8B	Magpie	54.34	44.81	68.77	44.05	25.10	47.41
	Reflection-tuning	57.08	66.49	67.32	50.34	30.09	54.26
	IT_0 + Cat. ($k = 0$)	57.60	64.59	70.14	49.23	31.46	54.60
	Length Selector						
	IT_1 + Cat. ($k = 1$)	58.31	65.73	72.87	48.05	34.08	55.81
	IT_2 + Cat. ($k = 2$)	58.27	69.22	72.70	51.99	32.03	56.84
	IT_3 + Cat. ($k = 3$)	57.91	67.25	72.70	52.88	34.06	56.96
	IT_4 + Cat. ($k = 4$)	57.27	67.17	72.87	49.83	33.44	56.12
	IT_5 + Cat. ($k = 5$)	56.17	65.73	69.45	50.27	33.75	55.07
	Off-policy Selector						
	IT_1 + Cat. ($k = 1$)	58.26	67.70	74.23	46.11	32.14	55.69
	IT_2 + Cat. ($k = 2$)	58.50	67.25	73.46	48.03	31.28	55.70
	IT_3 + Cat. ($k = 3$)	58.30	65.43	73.04	46.93	30.37	54.81
	IT_4 + Cat. ($k = 4$)	58.57	67.40	74.15	46.93	30.03	55.42
	IT_5 + Cat. ($k = 5$)	57.72	66.79	75.51	47.40	31.12	55.71
	On-policy Selector						
	IT_1 + Cat. ($k = 1$)	57.96	65.20	70.05	45.07	29.69	53.59
	IT_2 + Cat. ($k = 2$)	58.53	64.67	69.20	48.41	31.32	54.43
	IT_3 + Cat. ($k = 3$)	58.26	66.41	72.35	50.43	32.89	56.07
	IT_4 + Cat. ($k = 4$)	58.45	68.76	70.14	46.54	33.37	55.45
	IT_5 + Cat. ($k = 5$)	59.17	66.19	69.88	50.76	33.01	55.80

Table 7: The detailed experimental results of each iteration.

Data	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
Reflection-tuning	57.08	66.49	67.32	50.34	30.09	54.26
IT_1	57.74	67.93	74.06	49.22	32.94	56.38
IT_2	57.87	66.87	72.53	48.63	30.46	55.27
IT_3	57.51	68.46	73.21	51.59	31.61	56.48

Table 8: The results of Llama3.1-8B on instruction-tuning dataset (collected by length selector) of each iteration.

Data	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
SRLM – $\mathcal{M}_0$						
Cat. = 5k	58.47	67.10	69.54	47.13	27.98	54.04
Cat. = 10k	59.22	66.26	67.58	46.68	30.78	54.10
SRLM – $\mathcal{M}_1$						
Cat. = 5k	58.04	66.41	75.17	45.71	31.47	55.36
Cat. = 10k	58.56	66.72	75.00	48.00	30.02	55.66
SRLM – $\mathcal{M}_2$						
Cat. = 5k	58.05	68.46	72.70	44.76	29.28	54.65
Cat. = 10k	57.84	67.70	73.04	50.79	28.70	55.61
SRLM – $\mathcal{M}_3$						
Cat. = 5k	57.35	65.96	73.04	45.52	30.73	54.52
Cat. = 10k	57.29	67.32	73.38	48.65	29.23	55.17

Table 9: The results using different size of reasoning catalyst data based on Llama3.1-8B model with 3 epochs optimization on three iterations ($\mathcal{M}_{0,1,2,3}$). We **bold** the best performance at each iteration.

Data	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
IT_0 + Cat.	57.60	64.59	70.14	49.23	31.46	54.60
Update r_j^* *with* r_j^{t+1}						
IT_1 + Up. Cat._1	59.27	66.03	74.15	49.84	31.29	56.12
IT_2 + Up. Cat._2	57.42	65.81	71.08	47.57	29.45	54.27
IT_3 + Up. Cat._3	55.52	65.13	69.28	49.25	30.92	54.02
Update both r_j^0, r_j^* *with* r_j^t, r_j^{t+1}						
IT_1 + Up. Cat._1	56.99	64.75	72.87	47.94	33.83	55.28
IT_2 + Up. Cat._2	57.18	63.99	67.92	49.78	33.11	54.40
IT_3 + Up. Cat._3	55.09	65.13	68.17	49.42	32.57	54.08

Table 10: The performance of SRLM with length selector on Llama3.1-8B using *Update* 1k reasoning catalyst data with 1000 steps optimization.

# Sampling	**Methods**	MMLU	GSM8K	ARC-C	HellaSwag	BBH	Avg.
1	Reflection-tuning	57.08	66.49	67.32	50.34	30.09	54.26
	SRLM	57.91	67.25	72.70	52.88	34.06	56.96
2	Reflection-tuning	65.50	74.07	76.02	60.88	41.55	63.60
	SRLM	69.56	75.89	83.02	64.25	46.61	67.87
4	Reflection-tuning	71.66	80.52	82.00	70.33	52.48	71.40
	SRLM	78.44	83.85	89.51	76.11	57.74	77.13
8	Reflection-tuning	76.42	84.38	84.56	77.81	61.09	76.85
	SRLM	85.41	88.32	93.94	85.46	68.23	84.27
16	Reflection-tuning	79.79	88.10	85.75	82.61	67.56	80.76
	SRLM	90.31	91.89	96.33	91.67	74.87	89.01
32	Reflection-tuning	82.48	90.75	88.05	86.41	72.79	84.10
	SRLM	93.85	93.63	98.12	95.41	80.23	92.25
64	Reflection-tuning	84.80	92.27	89.16	89.24	76.27	86.35
	SRLM	96.15	95.38	98.72	97.62	83.31	94.24

Table 11: The detailed experimental results with multiple sampling for baseline and SRLM.

Question: Suppose a and b are positive integers, neither of which is a multiple of 3. Find the least possible remainder when $a^2 + b^2$ is divided by 3.

Reflection-tuning Answer: To solve this problem, we need to find the remainder of $a^2 + b^2$ when divided by 3, given that a and b are integers not divisible by 3.
First, consider the properties of numbers modulo 3. Any integer n can be expressed as $n = 3k + r$ where r is the remainder when n is divided by 3. Since a and b are not multiples of 3, $r \neq 0$.
Now, let's express a and b in terms of their remainders when divided by 3: $a = 3m + r_1$ and $b = 3n + r_2$, where $r_1, r_2 \neq 0$.
Substitute these into $a^2 + b^2$: $a^2 + b^2 = (3m + r_1)^2 + (3n + r_2)^2 = 9m^2 + 6mr_1 + r_1^2 + 9n^2 + 6nr_2 + r_2^2$.
Notice that all terms containing m and n are multiples of 3, and thus their sum is also a multiple of 3. The only terms not divisible by 3 are r_1^2 and r_2^2.
Since r_1 and r_2 are not zero, r_1^2 and r_2^2 can each be 1 or 2 (since they are squares of non-zero remainders when divided by 3). The smallest possible value for $r_1^2 + r_2^2$ is 1 (when both r_1 and r_2 are 1 or both are 2) and the largest possible value is 4 (when one is 1 and the other is 2).Thus, the remainder of $a^2 + b^2$ when divided by 3 can only be 1 or 2. , the possible remainders when $a^2 + b^2$ is divided by 3 are $\boxed{1, 2}$.

SRLM: <thoughts><decomposition> Let's break down the problem into smaller parts to ensure clarity and accuracy in our reasoning. First, we need to understand the behavior of squares modulo 3. Next, we'll examine the possible values of a and b under the constraint that they're not multiples of 3. Finally, we'll compute ($a^2 + b^2 \mod 3$) for these values and determine the remainder. </decomposition>
<detail> Understanding the behavior of numbers under modulo 3 involves recognizing that any integer n can be expressed as n = 3q + r, where q is the quotient and r is the remainder. For n^2, we have $n^2 = (3q+r)^2 = 9q^2+6qr+r^2$. Simplifying gives us $n^2 \equiv r^2 \pmod 3$. Thus, the possible remainders for n^2 are 0, 1, or 2 depending on r. </detail>
<detail> Since neither a nor b is a multiple of 3, they can take values of 1 or 2 modulo 3. This means $a \equiv 1 or 2 \pmod 3$ and similarly for b. We need to evaluate $a^2 + b^2$ for these possibilities. </detail>
<check> Let's verify each case:- Case 1: $a \equiv 1 \pmod 3, b \equiv 1 \pmod 3$- Then $a^2 \equiv 1^2 \equiv 1 \pmod 3$- $b^2 \equiv 1^2 \equiv 1 \pmod 3$- So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$
- Case 2: $a \equiv 1 \pmod 3, b \equiv 2 \pmod 3$- Then $a^2 \equiv 1^2 \equiv 1 \pmod 3$- $b^2 \equiv 2^2 \equiv 4 \equiv 1 \pmod 3$- So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$
- Case 3: $a \equiv 2 \pmod 3, b \equiv 1 \pmod 3$- Then $a^2 \equiv 2^2 \equiv 4 \equiv 1 \pmod 3$- $b^2 \equiv 1^2 \equiv 1 \pmod 3$- So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$
- Case 4: $a \equiv 2 \pmod 3, b \equiv 2 \pmod 3$- Then $a^2 \equiv 2^2 \equiv 4 \equiv 1 \pmod 3$- $b^2 \equiv 2^2 \equiv 4 \equiv 1 \pmod 3$- So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$</check>
<summary> In all cases, the remainder when $a^2 + b^2$ is divided by 3 is consistently 2. There were no overlooked scenarios or computational errors, confirming the correctness of the result. </summary>
<reflection> By systematically evaluating each case and cross-verifying calculations, I ensured that the solution aligns with the conditions provided. The consistency in outcomes confirms the robustness of the method applied. </reflection>
Therefore, the least possible remainder when $a^2 + b^2$ is divided by 3 is $\boxed{2}$.</thoughts>
Based on the above processing, the key reasoning steps can be summarized clearly, followed by the concise answer.solve this problem, we need to understand the behavior of squares modulo 3. Any integer n can be expressed as $n = 3q + r$, where q is the quotient and r is the remainder when n is divided by 3. Squaring both sides gives $n^2 = (3q + r)^2 = 9q^2+6qr+r^2$. This simplifies to $n^2 \equiv r^2 \pmod 3$. Therefore, the possible remainders for n^2 are 0, 1, or 2 depending on r.
Since neither a nor b is a multiple of 3, they can be either 1 or 2 modulo 3. We need to calculate a^2+b^2 for these cases: 1. If $a \equiv 1 \pmod 3$ and $b \equiv 1 \pmod 3$, then $a^2 \equiv 1^2 \equiv 1 \pmod 3$ and $b^2 \equiv 1^2 \equiv 1 \pmod 3$. So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$.2. If $a \equiv 1 \pmod 3$ and $b \equiv 2 \pmod 3$, then $a^2 \equiv 1^2 \equiv 1 \pmod 3$ and $b^2 \equiv 2^2 \equiv 1 \pmod 3$. So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$.3. If $a \equiv 2 \pmod 3$ and $b \equiv 1 \pmod 3$, then $a^2 \equiv 2^2 \equiv 1 \pmod 3$ and $b^2 \equiv 1^2 \equiv 1 \pmod 3$. So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$.4. If $a \equiv 2 \pmod 3$ and $b \equiv 2 \pmod 3$, then $a^2 \equiv 2^2 \equiv 1 \pmod 3$ and $b^2 \equiv 2^2 \equiv 1 \pmod 3$. So, $a^2 + b^2 \equiv 1 + 1 \equiv 2 \pmod 3$.all cases, the remainder when $a^2 + b^2$ is divided by 3 is 2. Therefore, the least possible remainder is $\boxed{2}$.

Table 12: The (non-cherry-pick) test instance from MATH.